National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

THE UNIVERSITY OF ALBERTA

CONTROL STRATEGIES FOR ROBOT ARM MOTION

by

GARRY MICHAEL STEIL

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE in CONTROL SYSTEMS

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1987

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-40972-X

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR GARRY MICHAEL STEIL

TITLE OF THESIS CONTROL STRATEGIES FOR ROBOT ARM MOTION

DEGREE FOR WHICH THESIS WAS PRESENTED MASTER OF SCIENCE

YEAR THIS DEGREE GRANTED FALL 1987

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) .....

PERMANENT ADDRESS:

#104 10745-78 Ave

Edmonton, Alberta

T6E-1P7

DATED October 21 1987

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled CONTROL STRATEGIES FOR ROBOT ARM MOTION submitted by GARRY MICHAEL STEIL in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE in CONTROL SYSTEMS.

V. G. Gourishankar

.....

Supervisor

R. E. Rink

.....

Rozin W. Toogood

.....

Date June 23 1987

ABSTRACT

In this thesis, a multi-phase control strategy is developed for controlling robotic arm movement. The proposed controller is designed to govern the three phases of a typical robot task - free motion, guarded motion, and compliant motion. This results in three separate levels of control. Use of classical proportional - integral - derivative strategies form the basis for the design. The derivation of the design is then enhanced through the use of linearized state-space modeling. The proposed designs are evaluated through simulation studies on a PUMA 600 robot arm.

To lay the groundwork for the simulation studies, a review of task planning, kinematics, and dynamics is first presented. A typical robot arm task which involves free, guarded, and compliant motion is described. The forward and inverse kinematic solutions for the PUMA robot are then developed. Working from a cartesian reference frame, the inverse kinematic solution is then utilized so as to provide reference values for position and velocity in the joint coordinate system. These values form the inputs to the servo motor controllers located at each joint.

Working from the Lagrangian formulation, the closed form dynamic equations of motion describing the Puma 600 are are symbolically generated. These equations are then combined with the dynamic equations describing the servo-motors forming an open-loop system.

From the open-loop system dynamics a PD controller is developed for use in the free motion phase. This controller is then modified for the guarded motion phase by including integral action. For the compliant phase, the controller is further modified by the addition of a cascaded force feedback loop. The design procedure yields fixed gain control for all three levels.

For more demanding tasks, feedforward techniques are utilized during the free motion phase. During the guarded and compliant phases, a linearized state-space controller with feedforward decoupling is proposed. This is seen to improve the overall speed at which the manipulator is allowed to operate.

ACKNOWLEDGEMENTS

The author acknowledges with deep gratitude the encouragement and support provided by his supervisor Prof. V. G. Gourishankar throughout the course of this work.

In addition, the author is deeply indebted to Dr. Leszek Trybus for his initial input and guidance in this area.

The support of the Department of Electrical Engineering and its graduate chairman Dr. R. P. W. Lawson is also gratefully noted.

Table of Contents

Chapter		Page
1.	Introduction	1
	1.1 Robotics Background	1
	1.2 Robotic Control	6
	1.3 Scope of Thesis	10
2.	Kinematics and Task Planning	13
	2.1 Kinematics	13
	2.2 Coordinate Transformations	14
	2.3 Task Planner	18
3.	Manipulator Dynamics	30
	3.1 Introduction	30
	3.2 Dynamic Equations of the Arm	32
	3.3 Servo-motor Dynamics	38
	3.4 Modified Manipulator Dynamics	40
4.	Fixed Gain Conventional Servo Design	43
	4.1 Introduction	43
	4.2 Servo Design Technique Based on a Simplified Model	45
	4.3 Free motion PD Control Utilizing a Full Dynamic Model	50
	4.4 Guarded Motion PID Control	52
	4.5 Compliant Motion Cascade Control	57
	4.6 Simulation Results	61
5.	State Space Controller Design Enhancements	66
	5.1 Introduction	66
	5.2 Free Motion State-space Design	67
	5.3 Guarded and Compliant Motion State-space Design	68
	5.4 Simulation Results	73

6. Conclusions 84
6.1 Review of Results 84
6.2 Future Work 86
REFERENCES 88
APPENDIX 1 93
APPENDIX 2 100
APPENDIX 3 114

List of Figures

Figure		Page
2.1	Puma 3-Link Manipulator	15
2.2	Free Motion Trajectory	19
2.3	Guarded and Compliant Motion Trajectory	20
2.4	Spatial Characteristics of Free Motion	23
2.5	Free Motion Temporal Characteristics	24
2.6	Free Motion Joint References	26
2.7	Guarded and Compliant Motion Temporal Characteristics	27
3.1	Inverse Dynamic Solution for the Free Motion Phase	34
4.1	Conventional PD Servo Control System	48
4.2	Guarded Motion PID Control Scheme: a) Block Diagram b) Root-locus	53
4.3	Compliant Motion Control Scheme: a) Block Diagram, b,c) Simplified block diagrams, d) Root-locus plot.	58
4.4	Free Motion Joint Reference Tracking	63
4.5	Guarded and Compliant Motion Cartesian Errors	64
5.1	Free Motion State Space Control	69
5.2	Guarded and Compliant Motiuon State Space Control	71
5.3	Root-locus for Cascade Control	74
5.4	Free Motion State Space Tracking with α=0.0	77
5.5	Free Motion State Space Tracking with α=1.0	78
5.6	Free Motion State Space Tracking with α=0.8	79
5.7	Free Motion State Space Tracking with α=0.6	80

Figure Page

5.8 Linearized Statespace Controller Tracking Errors for two poles located with ρ = 0.25 and the third pole located at twice this value 81

5.9 Linearized Statespace Controller Tracking Errors for two poles located with ρ = 0.20 and the third pole located at three times this value 82

5.10 Linearized Statespace Controller Tracking Errors for two poles located at -5.0 and the third pole located at four times this value 83

1. Introduction

1.1 Robotics Background

Robotics research is largely an interdisciplinary study. Manipulator design and actuation, motion planning and control, tactile sensing, and robotic vision may all be considered as separate technical areas. One of the driving forces behind research in these areas is the advancement of computer technology. This is especially true in such areas as robotic vision or control, where many of the computations have to be performed in real time. Essentially, as computers become faster and more powerful, they acquire the capability of handling the increasingly complex tasks associated with incorporating robots into more demanding environments. This promotes research into all areas associated with robotics. In this work, however, the planning and control of robotic motion is the primary focus.

Planning robotic motions may be considered to be one duty of a more broadly defined *task-planner*. A task-planner, by definition [1] transforms task level commands into manipulator level commands. Task level commands are all defined by the desired task and are independent of the manipulator. Typical tasks may involve pick - and - place operations, inserting a peg into a slot or spot welding. Examples of task level commands might be to move to specified cartesian location, grasp a peg, move to different location, and insert the peg into a slot. It is assumed that

the task planner has a detailed knowledge of both the task and the manipulator. In general, a manipulator may be described by its *kinematics* and *dynamics*. The manipulator *forward kinematics* is used to obtain the end-effector position as a function of the joint coordinates (angles in the case of rotational joints; displacements in the case of translational joints). Conversely, the *inverse kinematic* solution provides the joint coordinates values given the task coordinate values (typically cartesian). The manipulator *forward dynamics* describes the manipulator motion given the forces applied at each of the joints. *Inverse dynamics* provides the force required at each joint from a knowledge of the desired motion. Generally, the desired motion is only known in the task coordinate system. The task-planner must provide all the manipulator level commands in terms of joint positions, velocities, accelerations and torques. For the purposes of this thesis, it is assumed that there are direct current (DC) servo-motors located at each joint, and that joint positions and end-effector forces can all be measured.

A typical robot task is likely to involve three phases of motion [1] - free, guarded, and compliant. The free motion phase occurs after the arm has secured its payload. This motion typically involves a high degree of speed while at the same time allowing for path positional errors. During this phase, there is typically some specified *path tolerance*. Once the arm nears a target surface or object,

with which it is to make contact, the guarded motion phase begins. This motion is typically much slower and requires higher accuracy. The final phase, compliance, begins once contact is made with the target. During this phase, the forces applied at the end-effector have to be controlled. The accuracy at the contact point is often specified in terms of *repeatability*. Note that repeatability is a relative measure of accuracy rather than an absolute one.

For the free motion phase, the end-effector may or may not be constrained to follow a particular cartesian path. This depends on the nature of the task in so far as obstacles must be avoided. Should the end-effector not be constrained to follow a particular path, trajectories can be achieved in the joint coordinate system. The simplest joint coordinate trajectory planning would be to apply step reference changes to the control loops operating at each of the joint servos. The controller would then move the manipulator from its initial position to the desired position. The drawback with this scheme lies with the controller in that responses at the joint level must be overdamped [2]. This greatly reduces the speed at which the system is allowed to operate. Alternative joint level references have been investigated in the literature [3]. These include constant acceleration/deceleration profiles (approximate bang-bang), as well as cubic spline, and polynomial interpolations. For all such inputs, the arm dynamics and servo capabilities are largely ignored. That

is, there is no guarantee that the servo will have sufficient power to execute the desired motion. This demonstrates that the task planning problem cannot be solved independently from the dynamics problem.

In general, it should be noted that the inverse kinematic problem may not be unique. To overcome this problem, Whitney [4] has proposed minimizing a weighted sum of angular velocity terms. Alternative time-optimal path planning algorithms have been proposed for both joint [5] and cartesian [6] coordinates. These algorithms are based on minimum traveling times subject to globally maximum joint velocity and acceleration constraints. The resulting path positional references consist of cubic polynomial interpolations between a sequence of knot points (desired joint positions obtained from the inverse kinematics). It is interesting to note that cubic polynomials can be considered to be the path solution to a double integral plant that has been subject to a minimum energy cost function. This result is shown in Young [7], where the dynamic equations of motion describing an arm have been simplified so as to approximate the double integral plant.

Should the free motion phase be constrained to follow a particular cartesian reference trajectory, the path planning algorithms discussed so far will not work. Cartesian reference trajectories arise whenever straight line motion or a specific curvilinear motion is desired. In these cases the joint references are obtained from the geometry of the

task. Two approaches for this have been reported in the literature. The first, is to solve the inverse kinematic solution based on an arbitrary arm configuration (elbow up/elbow down etc.)[8]. This solution is then parameterized in time, based on maximum cartesian accelerations and decelerations. Here, an exact analytic procedure for determining the nature of the acceleration/deceleration profile has yet to be developed. The second method, known as "Whitney's resolved acceleration control" [8], involves specifying cartesian reference velocities and integrating the Jacobian transformation between the joint velocities and the desired cartesian velocities.

Once the end-effector begins the approach to the target, free motion ends and the guarded motion phase begins. This phase almost always involves a straight line cartesian reference. The motion usually requires a high degree of accuracy and is performed at low speeds. During all low speed movements, many of the velocity and acceleration dependent dynamic terms are greatly reduced and consequently their significance may have to be reevaluated.

The final phase, compliance, begins once the end-effector makes contact with a rigid body. During this phase, force measurements may be obtained and compared with desired values, so that any errors in force may be compensated for by adjustments at the individual joints. This is realized through a knowledge of the "Force Jacobian" [9], which may be calculated by the task planner, or

obtained experimentally.

1.2 Robotic Control

The common objective of all phases of path planning is to provide input reference values to the controllers operating at the individual joints. A control algorithm is then used to move the arm along the desired trajectory. The control algorithms are usually complicated by the large dimensionality and non-linear nature of the system dynamics. To date, four main control strategies have been investigated in the literature. These are: (i) solving the inverse dynamic problem, (ii) applying adaptive control, (iii) solving for an optimal control law based on a prespecified cost function, and, (iv) applications of classical Proportional-Integral-Derivative (PID) control schemes. These strategies are also commonly used in combination.

Inverse dynamic solutions yield the necessary torques, to be applied at each joint, from a knowledge of the robot arm dynamics. Although these algorithms are computationally intensive, several versions which are suitable for real-time applications have been proposed [10,11]. However, these strategies are sensitive to parameter inaccuracies and unmodeled static and dynamic friction. As well, the solutions assume perfectly rigid joints, and ignore the servo-motor dynamics. Measurements of applied torque are also very difficult to obtain. Should these measurements not be available, the calculated torques must be applied in an

open-loop manner. A further drawback to these schemes is that the arm dynamics are load dependent. This means that on-line identification must also be used for varying payload applications.

Numerous adaptive control strategies have been proposed in the literature [12,13,14]. These algorithms all use some form of recursive least squares (RLS) identification. In [12], parameter identification is only performed in the immediate vicinity of a nominal trajectory which has first been obtained through an inverse dynamic solution. For this case, the identified parameters represent any unmodeled dynamics or inaccuracies in the robot model parameters (masses, moment of inertia, etc.). Using the identified parameters, an optimal one - step - ahead feedback controller is utilized. Here, it should be noted that any control law could have been used to account for the modeling errors (e.g. PID). Of the other adaptive algorithms considered here, [13] minimizes a standard quadratic cost function while [14] utilizes a pole placement algorithm. In all cases, no assessment is made as to how well the RLS identification algorithm tracks the rapidly changing parameters associated with robot arm dynamics. In general, global stability is difficult to ascertain for RLS algorithms.

Optimal control strategies have roughly divided into two categories - linear quadratic (LQ), and time optimal with a weighted fuel criterion. In [15,16] the LQ problem

has been solved based on a simplified dynamic model. These solutions have been highly criticized, in the literature, as being sub-optimal. It should be noted that the general LQ optimal control problem requires that the control signal be determined so as to move the controlled system from a specified initial value to a specified final value. For the the control of a robotic manipulator then, the optimal path would be contained in the solution of the optimal control law. That is, given a specific cost function, and the manipulator dynamic constraint equation, the solution of the optimal path could theoretically be obtained. Generally, this is not done, rather, the control law that yields the optimal path is directly obtained. To date, this problem has not been solved for a full robot dynamic constraint equation. Initially, the problem would have had to be one of unconstrained manipulator motion. For constrained manipulator motion, the LQ solution has been obtained which yields optimal PID control gains [17]. This has been determined utilizing the full dynamics constraint equation. The proposed solution requires a complete knowledge of the manipulator's inverse dynamics and thus makes the proposed scheme computationally more intensive than applying the inverse dynamic solution directly (i.e the Matrix Riccati equation must be solved off line [18]). For step responses the authors suggest propagating the Riccati equation to steady state so as to obtain fixed gain PID control.

A time-optimal controller with weighted minimum-fuel criterion in proposed in [19]. This algorithm involves on-line identification of the dynamics. The manipulator end-effector is not constrained to follow any particular path and is therefore only applicable for unconstrained end-effector paths. This controller is essentially a free motion controller since steady state accuracy cannot be guaranteed. The authors acknowledge this limitation and recommend classical PI control be implemented near the end of the trajectory so as to ensure steady state accuracy.

To date, only a few articles dealing explicitly with classical PID control have appeared. Paul, in his well known book [2], describes PD design in a manner that is similar to that used in industry (single-input-single-output (SISO) control with some feedforward capabilities). In [8,20], a very detailed analysis of servo dynamics and arm inertias is given, along with recommended PD control designs. As well, extensive use of nonlinear feedforward (feeding forward partial inverse dynamic solutions) is investigated.

Most of the control literature dealing with robotics has concerned itself with the free motion phase of a robot task. For the guarded motion phase, most authors have relied on feedforward techniques to meet the high accuracy demands (both [2,20] use this approach). For the compliant motion, Raibert and Craig [21] have proposed a hybrid PID controller which controls either position or force but not both. That is, positional control is dropped along the force controlled

axes and force control is dropped along the position controlled axes.

In summary, adaptive control and optimal control strategies proposed in the literature tend to be compared to classical PID control, with each case showing drastic improvements in performance. However, the methodology used to "tune" the PID controller settings is seldom explicitly stated. This makes the comparisons more difficult to evaluate.

1.3 Scope of Thesis

This thesis concerns itself with the problem of tracking joint reference values for the free, guarded and compliant phases of a robot task. The joint reference values are obtained from a task planner that works entirely in cartesian coordinates (constrained end-effector motion). For the tracking problem, optimal and adaptive schemes are not considered. Rather, classical PID control is extended so as to account for the changing demands of the three phases. A detailed design methodology is presented.

Chapter 2 develops a task planner for the PUMA 600 robot. The forward and inverse kinematic solutions are presented. For the free motion, a cartesian curvilinear path with an approximate bang-bang cartesian temporal profile is proposed. The guarded phase involves tracking a cartesian straight line in the horizontal direction. This requires the arm to act against the gravity field. The compliant phase

involves inserting a peg into a hole. During this motion, both force and positional references are provided.

In Chapter 3, the dynamic equations of motion for the arm are developed following Neuman and Murray's symbolic algorithm [22]. By using a symbolic approach the significance of each dynamic term can be evaluated separately. This is essential since many control algorithms assume that the certain dynamic terms are insignificant. Certainly, for very slow motions, such as in guarded and compliant motion the velocity dependent dynamics may become small. Following the discussion of the dynamic equations, a linearized state-space model is obtained [23]. This model is essential to the control designs presented in chapter 5. Finally, a review of the well known equations describing a permanent magnet DC voltage controlled motor is presented.

In Chapter 4, free motion PD designs [2,8] are reviewed and design modifications presented. A "split-P-mode", guarded motion controller is then developed for use in the guarded motion phase. This controller emphasizes integral action as an alternative to the feedforward techniques proposed in [2] and [20]. The advantage of integral action is that zero steady state errors can be guaranteed without an accurate knowledge of the system parameters. A cascaded PID-force feedback controller is then proposed for the compliant phase. This design controls both position and force. All the designs are fixed gain as prefered by industry. The designs account for both the servo motor

dynamics as well as the arm dynamics.

Chapter 5 investigates control designs based on state space modeling. For the free motion phase, non-linear feedforward and state references are utilized. During guarded and compliant motion, the linearized model presented in chapter 3 is utilized. The linearized design is implemented during guarded and compliant motion only, since these phases are limited to a small range of motion. The design modifications are used to increase the speed at which the final two phases are allowed to operate.

Chapter 6 discuses general conclusions that may be derived from this work. Having developed a detailed design methodology for the design of classical controllers, and having evaluated these designs on what may be considered a very demanding task, future research involving optimal and adaptive designs is discussed.

2. Kinematics and Task Planning

2.1 Kinematics

Kinematics deals with the geometric description of the robot arm motion as a function of time, without actually considering the forces which would accomplish such motion. Determining such forces is treated separately and is known as the *dynamics* problem. The direct, or forward *kinematic* problem is to obtain the end-effector's position, in the base cartesian coordinate system, from a knowledge of the individual joint positions. The inverse kinematic problem is to obtain the joint position given the end-effector's cartesian coordinates. The inverse problem may not always have a unique solution (degeneracy).

The end-effector trajectory may be obtained in a number of ways. One such way, popular in industry, is to trace the desired path through by hand and have a computer quantize the resulting joint angles (manual-lead-through). As an alternative to this, the desired cartesian path may be provided by the user.

Although the kinematics problem is treated separately from the dynamics problem, the two are not independent. That is, the trajectory planner must not put out joint references that cannot be realized by the available servo power. For the trajectory planner to know if it has exceeded the bounds on servo power it must calculate the inverse dynamics [10,11].

2.2 Coordinate Transformations

To solve the forward kinematic problem, a coordinate system is formed for each link. Homogeneous transformation matrices are then used to transform vectors from one coordinate system to another. For this purpose, the Denavit-Hartenberg (D-H) convention [1,2,8,9,32] has become widely accepted. This convention establishes rules as to where the coordinate system for each link is to be located, and provides the required transformations.

Shown in Figure 2.1 is a three link Puma robot and its base cartesian coordinate system. The corresponding robot link parameters as established by the D-H convention are given in Table 1.

TABLE 1
Link Coordinate System For The PUMA 600 Robot

Joint i	d_i (m)	a_i (m)	α_i (deg)	Range (deg)
1	0	0	-90	-160 to 160
2	0.1495	0.4320	0	-225 to 45
3	0	0	90	-45 to 225

The homogeneous transformation A_n, which transforms points from the n^{th} coordinate system to the $(n-1)^{th}$ coordinate

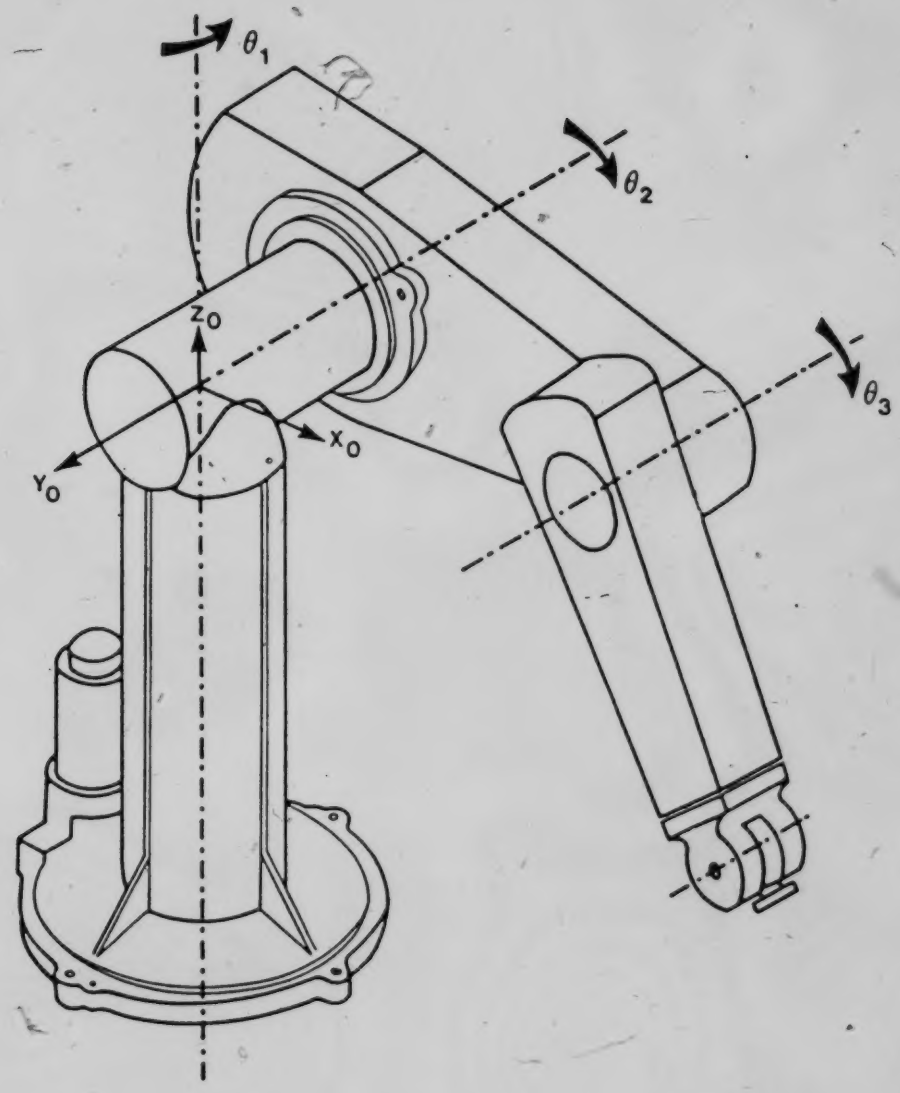


Figure 2.1 Puma 3-Link Manipulator

system, is given as:

$$A_n = \begin{bmatrix} C_n & -\cos(\alpha_n)S_n & \sin(\alpha_n)S_n & a_nC_n \\ S_n & \cos(\alpha_n)C_n & -\sin(\alpha_n)C_n & a_nS_n \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where C_i and S_i represent $\cos(\theta_i)$ and $\sin(\theta_i)$. From Table 1, the individual transformations for the PUMA are obtained as:

$$A_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From the homogeneous transformations, any vector in the third coordinate frame may be located in the base coordinate

frame by the transformation:

$$T_{03} = A_1A_2A_3$$

$$\begin{bmatrix} C_1C_{23} & -S_1 & C_1S_{23} & (a_2C_1C_2 - d_2S_1) \\ S_1C_{23} & C_1 & S_1S_{23} & (a_2S_1C_2 + d_2C_1) \\ -S_{23} & 0 & C_{23} & -a_2S_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where C_{ij} and S_{ij} represent $\cos(\theta_i + \theta_j)$ and $\sin(\theta_i + \theta_j)$ respectively. The end effectors position in the third coordinate frame is located at $[\ 0\ 0\ d_4\ 1\ ]^T$. Premultipling this vector by T_{03} yields the direct kinematic equations for the end effector's cartesian location:

$$\begin{aligned} x &= C_1(a_2C_2 + d_4S_{23}) - d_2S_1 \\ y &= S_1(a_2C_2 + d_4S_{23}) + d_2C_1 \\ z &= d_4C_{23} - a_2S_2 \end{aligned} \qquad (2.1)$$

From the direct kinematic equations, the inverse solution is obtained by solving for $[\theta_1,\theta_2,\theta_3]$. This results in:

$$\theta_1 = \tan^{-1}\left[\frac{\pm y\sqrt{x^2 + y^2 - d_2^2} - d_2 x}{\pm x\sqrt{x^2 + y^2 - d_2^2} + d_2 y}\right] \qquad (2.2a)$$

$$\theta_3 = \tan^{-1}\left[\frac{x^2 + y^2 + z^2 - a_2^2 - d_4^2 + d_2^2}{\pm\sqrt{4a_2^2 d_4^2 - (x^2 + y^2 + z^2 - a_2^2 - d_4^2 - d_2^2)^2}}\right] \qquad (b)$$

$$\theta_2 = \tan^{-1}\left[\frac{-z(a_2 + d_4 S_3) \pm d_4 C_3\sqrt{x^2 + y^2 - d_2^2}}{(a_2 + d_4 S_3)\sqrt{x^2 + y^2 - d_2^2} + d_4 C_3 z}\right] \qquad (c)$$

For θ_3, the positive square root corresponds to the elbow-down configuration while the negative square root corresponds to the elbow-up configuration. For θ_1 and θ_2 the plus/minus square roots correspond to right and left arm configurations.

2.3 Task Planner

Figure 2.2 shows the free motion phase for a typical assembly line task that may be performed by a Puma 600 robot. The numerical values given are provided by the task planning algorithm. The free motion consists of picking up a part and rotating 90° about the waist (θ_1). Once this motion is completed, the arm enters the guarded motion phase as shown in Figure 2.3. This phase involves tracking a straight line in cartesian coordinates for about 2.5 centimeters at

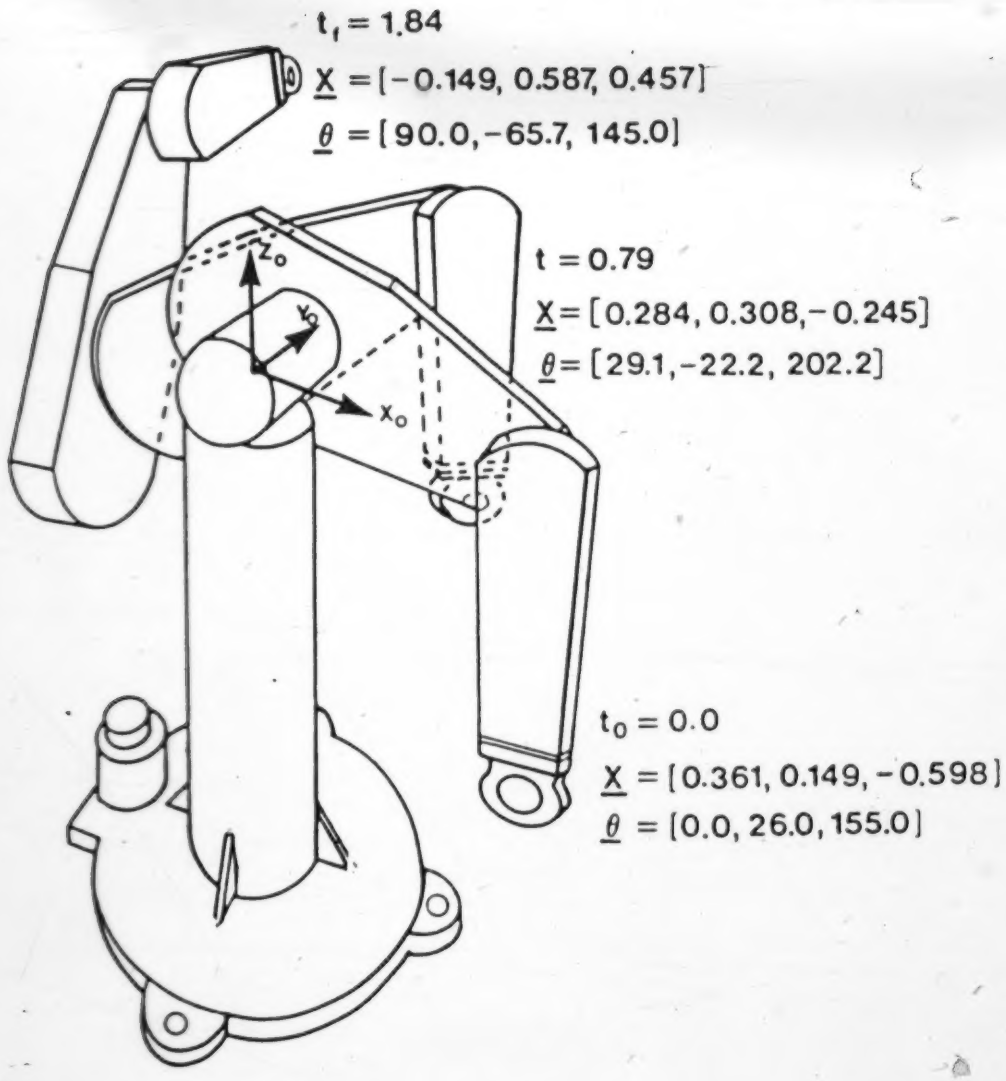


Figure 2.2 Free Motion Trajectory

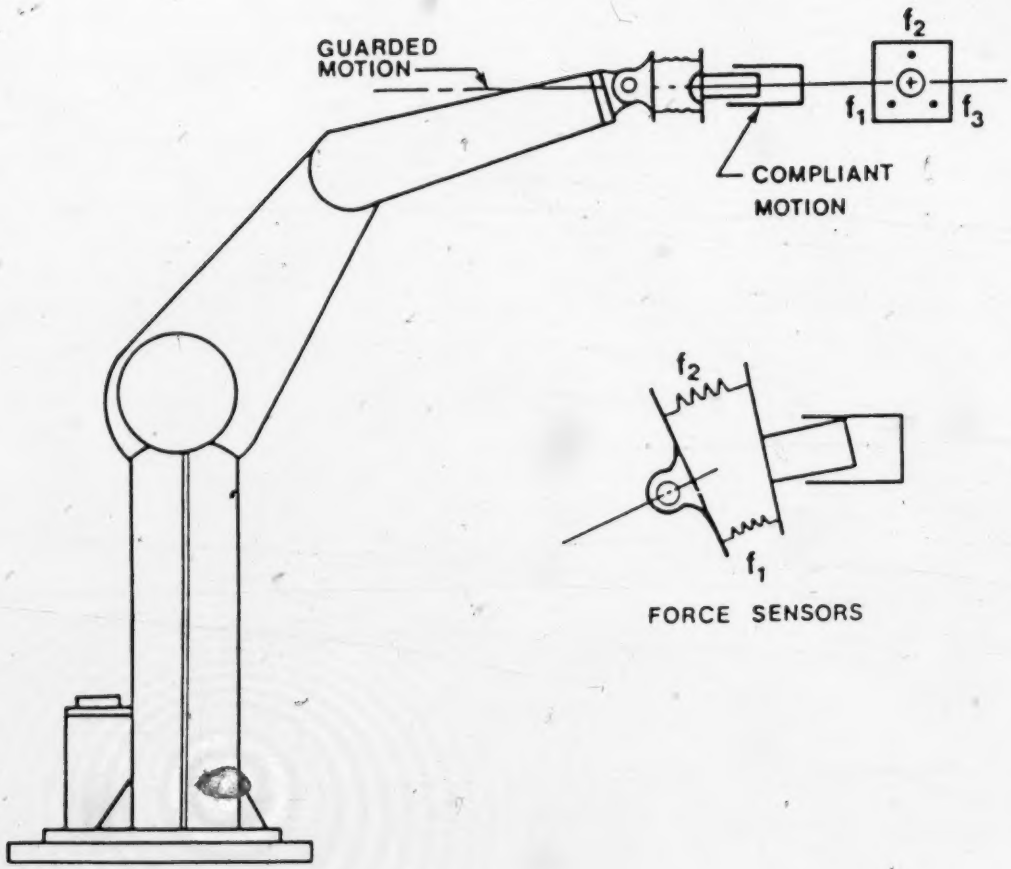


Figure 2.3 Guarded and Compliant Motion Trajectory

which time the arm makes contact with the surface and compliant motion begins. The compliant motion continues along the same straight line for a further 7.5 centimeters. This task is similar to pressing a bearing onto a shaft [9], as may be found on many assembly lines. Force sensors are available as shown in Figure 2.3 (inset). Consequently, it can be seen that for the limited range of compliant motion, both force and position references are available. For the compliant motion, it can be seen that the positional reference θ_1 is a constant (90°) while θ_2 and θ_3 will correspond to the straight cartesian line. For simplicity, the last three links are assumed to be locked rigid in the correct orientation. For a full 6-link PUMA, the kinematics and dynamics for all 6-links would have to be accounted for. In general though, the orientation problem is treated some what separately from the positional problem. As was the case with kinematics and dynamics, these two problems are not totally independent.

To obtain the numerical values shown in Figure 2.2, the free motion trajectory is parameterized by a circular arc in the plane coincidental with links 2 and 3 as shown in Figure 2.4. Such a trajectory would be typical for a robot that has been put through the desired motion by hand ("manual - lead - through"). The reference trajectory, in cartesian coordinates, is obtained from figure 2.4 as:

$$
\begin{aligned}
x_{ref} &= (R_1 - R_2\cos\phi(t))\sin\theta_1(t) + d_2\cos\theta_1(t) \\
y_{ref} &= (R_1 - R_2\cos\phi(t))\cos\theta_1(t) - d_2\sin\theta_1(t) \\
z_{ref} &= -R_3 + R_2\sin\phi(t) \\
\theta_1(t) &= \theta_1(t_0) + (\theta_1(t_f) - \theta_1(t_0)) \frac{(\phi(t) - \phi(t_0))}{(\phi(t_f) - \phi(t_0))}
\end{aligned}
\tag{2.3}
$$

Here, θ_1 refers to the joint angle and d_2 is the Denavit-Hartenburg joint parameter (offset distance between joint one's coordinate origin and joint two's coordinate origin measured along link 1). The variables t_0 and t_f denote initial and final times for the motion. For the task shown, R_1 = 2.246 m, R_2 = 1.89 m, and R_3 = 0.45 m.

The temporal characteristics of $\phi(t)$ and $\dot{\phi}(t)$ are shown in Figure 2.5. This trajectory is characterized near maximum accelerations and decelerations in ϕ. The accelerations and decelerations chosen have been optimized by numerical trial and error since a general optimizing scheme does not exist for the continuous inverse kinematic transformation. In general, the optimal acceleration profile for $\phi(t)$ would be an arbitrary function. Here, optimization is interpreted as choosing the time dependence ϕ so as to utilize the maximum servo power, given the desired cartesian path (i.e. the path is determined not for optimallity but rather for obstacle avoidance) For the chosen temporal characteristic the coriolis torques and cross coupled inertia torques prove to

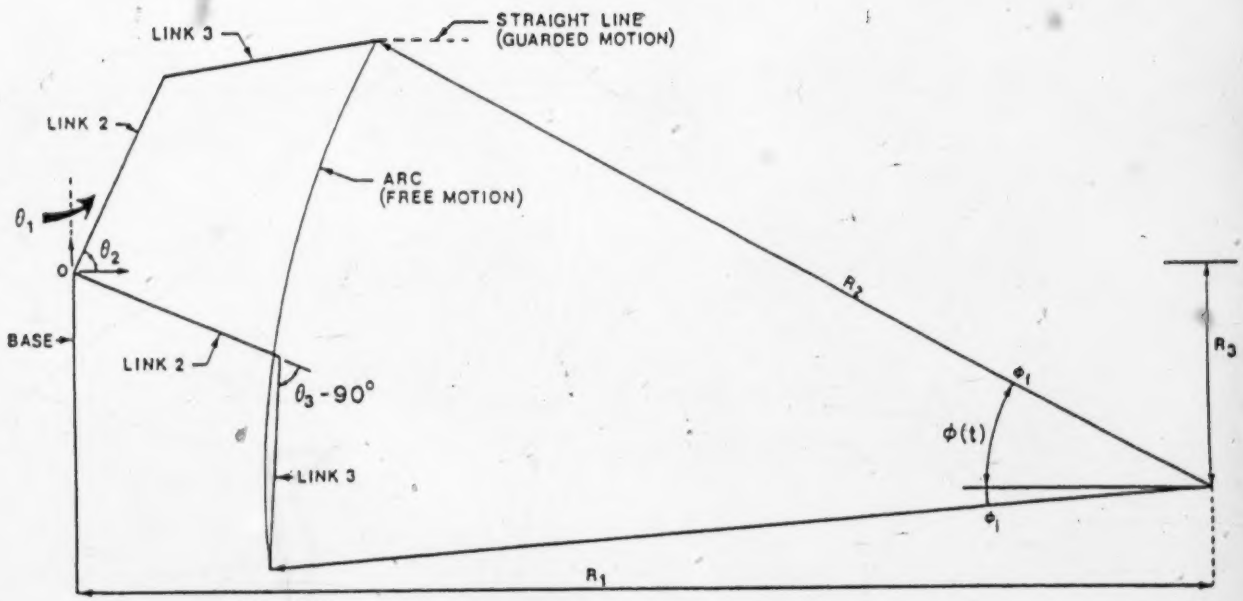


Figure 2.4 Spatial Characteristics of Free Motion

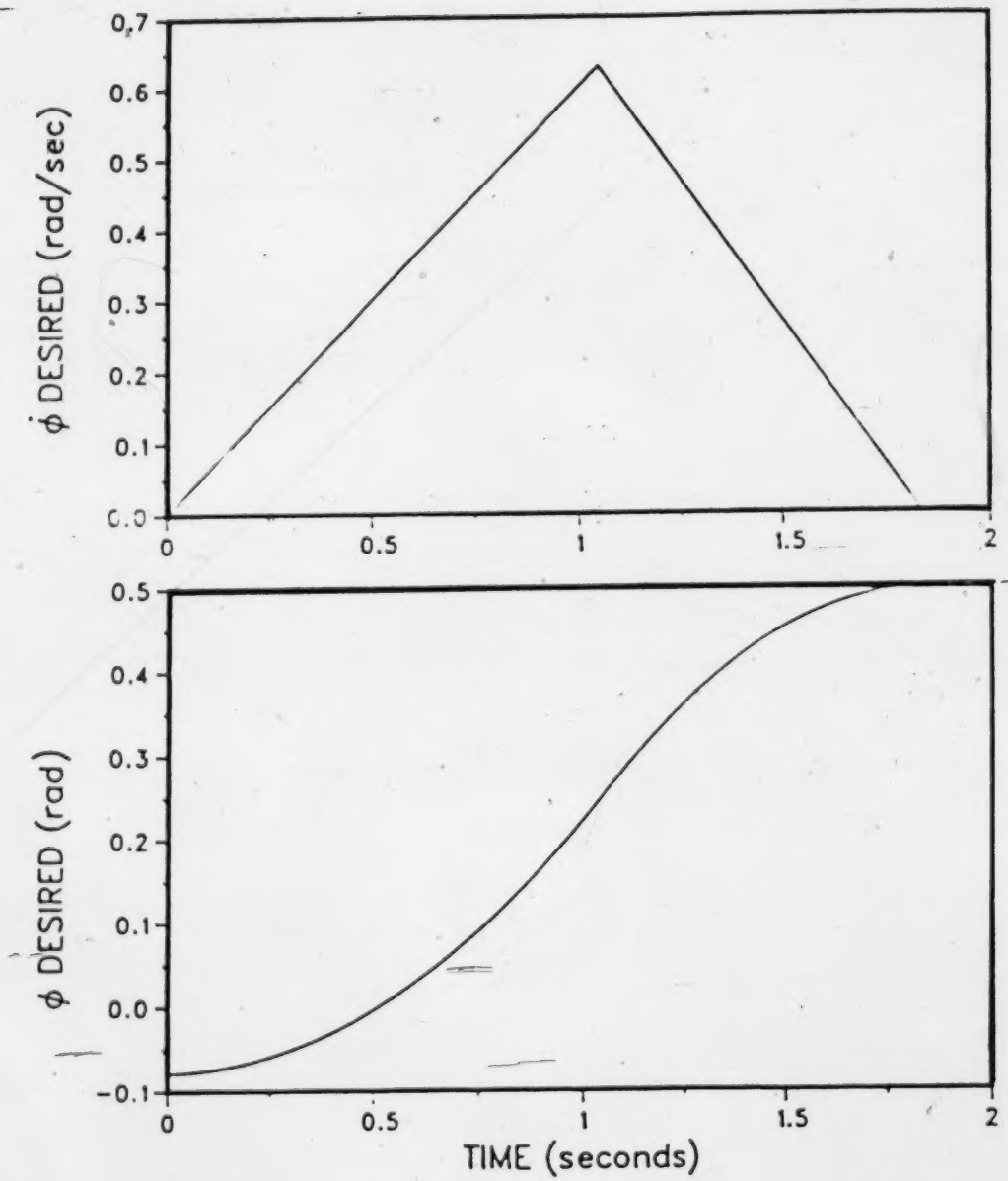


Figure 2.5 Free Motion Temporal Characteristics

be quite significant (same order of magnitude as the gravity torques).

From the cartesian reference trajectory, (equation 2.3) the desired joint angles $[\theta_1(t), \theta_2(t), \theta_3(t)]$ may be obtained from the inverse kinematics solution (equation 2.2). These references are shown in Figure 2.6. In general, continuous time transformations for the joint references are computationally intensive and are often replaced with linear piecewise approximations [9]. For the guarded and compliant phases, x(t) and z(t) are constant, with y(t) and y(t) as shown in Figure 2.7.

Should the end-effector not be constrained to follow any particular path, as in an obstacle free environment, any reference of the form (2.4) may be used (i.e. joint interpolated control).

$$\theta_{ref}(t) = \theta_i + (\theta_f - \theta_i)P(t) \qquad (2.4)$$

where $P(t) \in \{0,1\}$

Here, θ_i is the initial value of a particular joint and θ_f is the desired final value. Defining P(t) as a quadratic polynomial in time allows the user to explicitly input the joint velocities and accelerations. The advantage to this is that fast (high gain) controllers can be used with slow references (low frequency). This same advantage holds true for the joint references given in Figure 2.6. In [3],

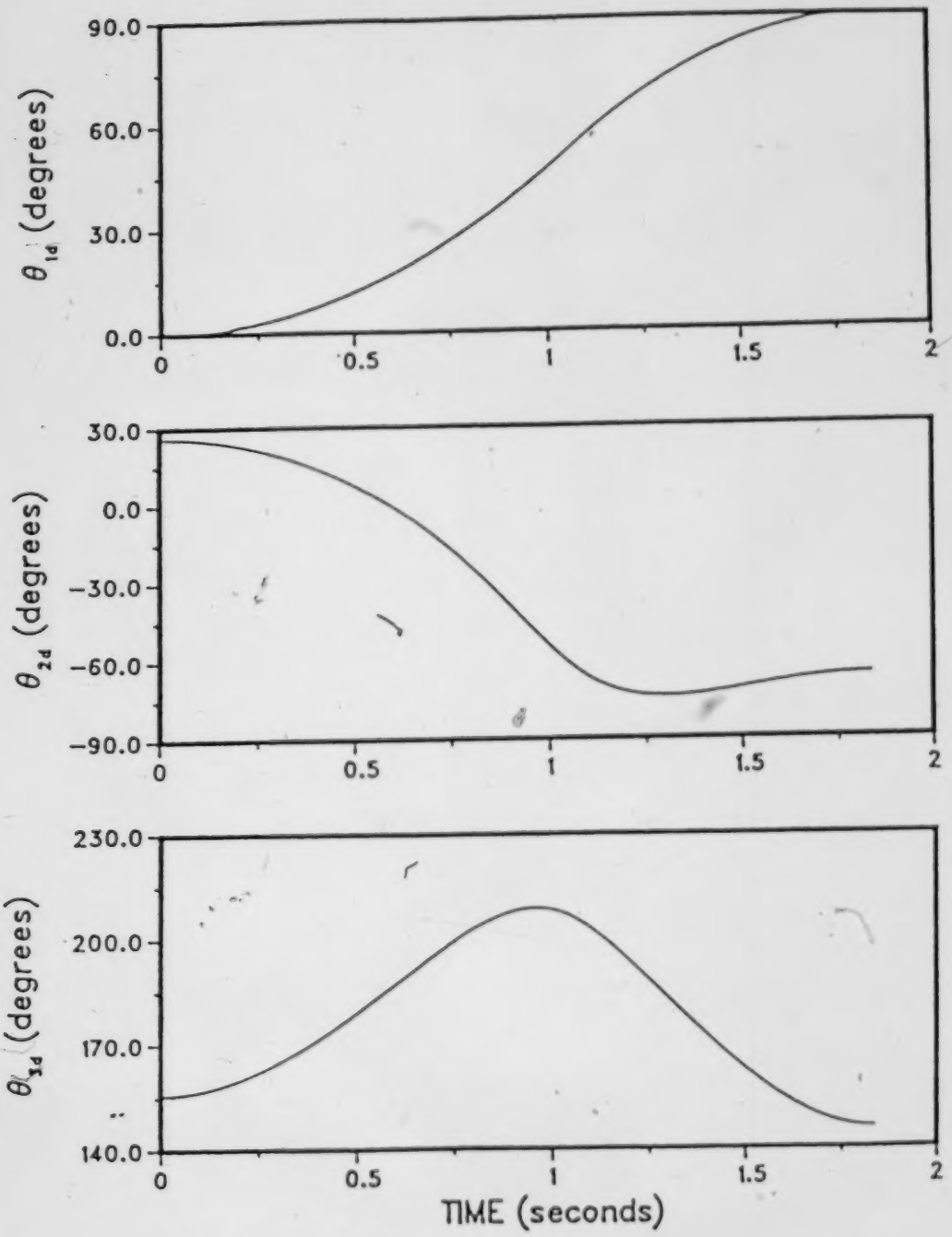


Figure 2.6 Free Motion Joint References

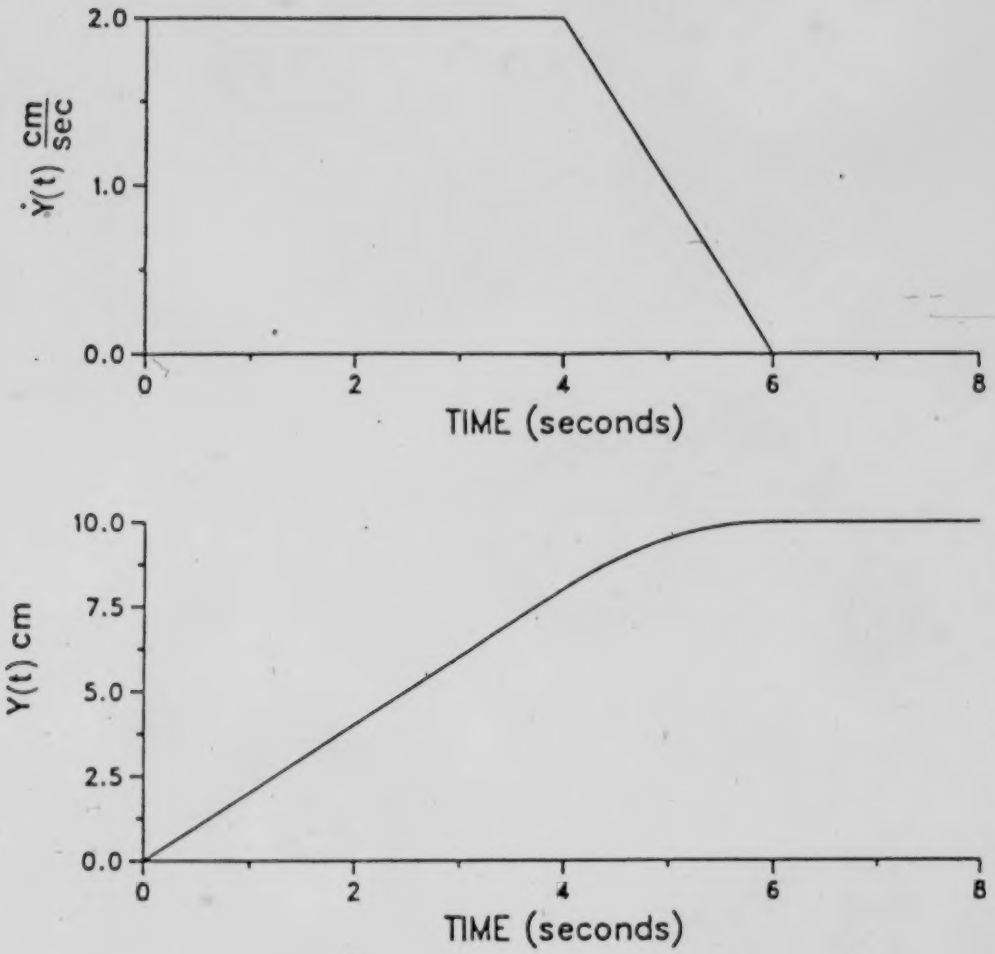


Figure 2.7 Guarded and Compliant Motion Temporal Characteristics

numerous forms of (2.4) are investigated. These authors choose their temporal dependence based on keeping the frequency spectrum of the joint reference below the structural resonant frequency of the robot [2]. Structural resonant frequency is discussed in more detail in Chapter 4.

For the part to be inserted properly (i.e. straight), all of the reference forces (f_1,f_2,f_3 in fig. 1b) will have to be equal. In general, the force exerted can be expressed as a function of θ:

$$f = f(\theta) \tag{2.5}$$

where f is the compliant force vector and θ is the position vector. From (2.5), the variations in force as a function of θ are obtained as:

$$\Delta f(\theta) = J(\theta)\Delta\theta \tag{2.6}$$

where the force Jacobian $J(\theta)$ is obtained as

$$J(\theta) = \frac{\partial f}{\partial \theta} \tag{2.7}$$

Equation (2.6) demonstrates that the desired force and position references are explicitly related. From the force Jacobian, imbalances in force may be compensated by

adjusting the joint angles as shown in equation 2.8.

$$\Delta\theta = J(\theta)^{-1}\Delta f \tag{2.8}$$

Here, Δf is the measured force imbalance. Note that a reference force is not explicitly required, although one may be provided. For a small range of motion, the force Jacobian need be calculated only at the point of contact. In general, $\Delta\theta$ obtained utilizing force measurement, would be the same as the joint positional errors if both the arm's position and the fixture's (i.e hole) position were known without error. For this case, the positional error and force error form a redundant set.

3. Manipulator Dynamics

3.1 Introduction

The dynamic equations describing a manipulator motion are described mathematically as a set of second order coupled nonlinear differential equations. A detailed knowledge of these equations is important to the development of control strategies in so far as they provide a priori information necessary for improving adaptive control algorithms, yield the constraint equations necessary for generating realistic optimal control designs, and generate valuable insight into selecting classical PID control gains. Essentially, if an arm were modeled without errors, the inverse dynamic solution would eliminate the need for any type of feedback control. Alternatively, if it were only required to move the manipulator from one position to another, control strategies involving integral action would eliminate the need for any detailed knowledge of the manipulator dynamics (here, some dynamic knowledge would be needed to insure stability). Thus, the relationship between the inverse dynamics and the feedback controller is not unique. It has been proposed in [12] that feedback control be used only to compensate for errors in position which result from inaccuracies in the inverse dynamic solution. Less intensive would be feedforward schemes, such as those presented in [1,20]. Feedforward may be considered as a partial inverse dynamic solution. The dynamics of a robot

arm, are of course necessary for the simulation and analysis of various control strategies.

Two approaches, Newton-Euler and Lagrangian, have been used to derive the dynamic equations describing multiple link manipulator motion. Both yield the same results but are computationally very different. Fast recursive formulations for both methods have been presented in the literature [10,11]. Presented here, is the closed form Lagrangian formulation. This form is not suitable for real-time inverse dynamic calculations since it is numerically inefficient (non-recursive). It is, however, practical in terms of the symbolic generation of the dynamic equations. The symbolic equations are necessary for feedforward calculations and are useful in investigating the functional dependency of load parameters on the dynamics. Furthermore, the symbolic dynamic equations lend themselves to obtaining closed form controller gains and lead directly to obtaining the linearized statespace model. This model is essential since most control theory is centered around linear systems.

The dynamic formulations discussed so far all assume rigid link construction (flexible link manipulators are not considered here). Non-linearities such as friction (static and dynamic), and gear backlash are not considered. The dynamics associated with the actuators are also largely ignored.

3.1 Dynamic Equations of the Arm

The general N-link dynamic equation of motion, describing a robot arm as obtained from the Lagrangian or Newton-Euler formulation [1] is:

$$T(t) = D(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) \qquad (3.1)$$

where $T(t)$ is the torque vector (N x 1)
$D(\theta)$ is the inertial coupling matrix (N x N)
$D(\theta)\ddot{\theta}$ is the inertial force vector (N x 1)
$C(\theta,\dot{\theta})$ is a coriolis force vector (N x 1)
$G(\theta)$ is the gravity loading vector (N x 1)

In analyzing these equations, it is useful to note the functional dependency of each of the terms since algorithms involving on-line identification could be used to identify the unknown, but *constant*, load parameters. Here, a priori knowledge of the link parameters and dynamics could be used to speed convergence.

To generate the individual terms in equation 3.1, the symbolic language REDUCE2 [25] is utilized. The symbolic algorithm is given in [21]. Program code for the PUMA 600 is given in Appendix 1 with the actual dynamic equations listed in the inverse dynamic software (Appendix 2).

The relative significance of inertial, coriolis, and gravity torques is very much task dependent. Of particular interest is the coriolis torque vector. Since this term is velocity dependent, many designers assume a priori that a

given manipulator will not have sufficient servo-power to achieve the necessary speeds to make this term significant when compared to the gravity and inertial torques [1].

For tasks that have been explicitly designed with free, guarded and compliant motion phases, it may be assumed that since the final two phases are performed at very low speeds, the coriolis torque will indeed be small. However, for the free motion phase, the coriolis torques could become quit significant if rapid motions are required. For the task shown in Figure 2.2, the inertial, coriolis, and gravity torques are all compared to the total torque required to perform the free motion (see Figure 3.1). From Figure 3.1, it can be seen that the coriolis force vector may indeed prove to be a significant disturbance seen by the feedback controller. Should the manipulator speed be significantly increased, these forces would also greatly increase (i.e. the coriolis torques are roughly proportional to the velocity squared). Thus, for rapid free motions, coriolis feed-forward may be beneficial.

For manipulator tasks acting against gravity, feedforward of the gravity torque may significantly decrease the settling time required. Essentially, for a PID control law the final value obtained from the I-mode would be equal to the gravity torque at that location. The proportionality constant associated with this mode would determine how fast the integrater reaches this value. However, a large proportionality constant may not be desired here, since this

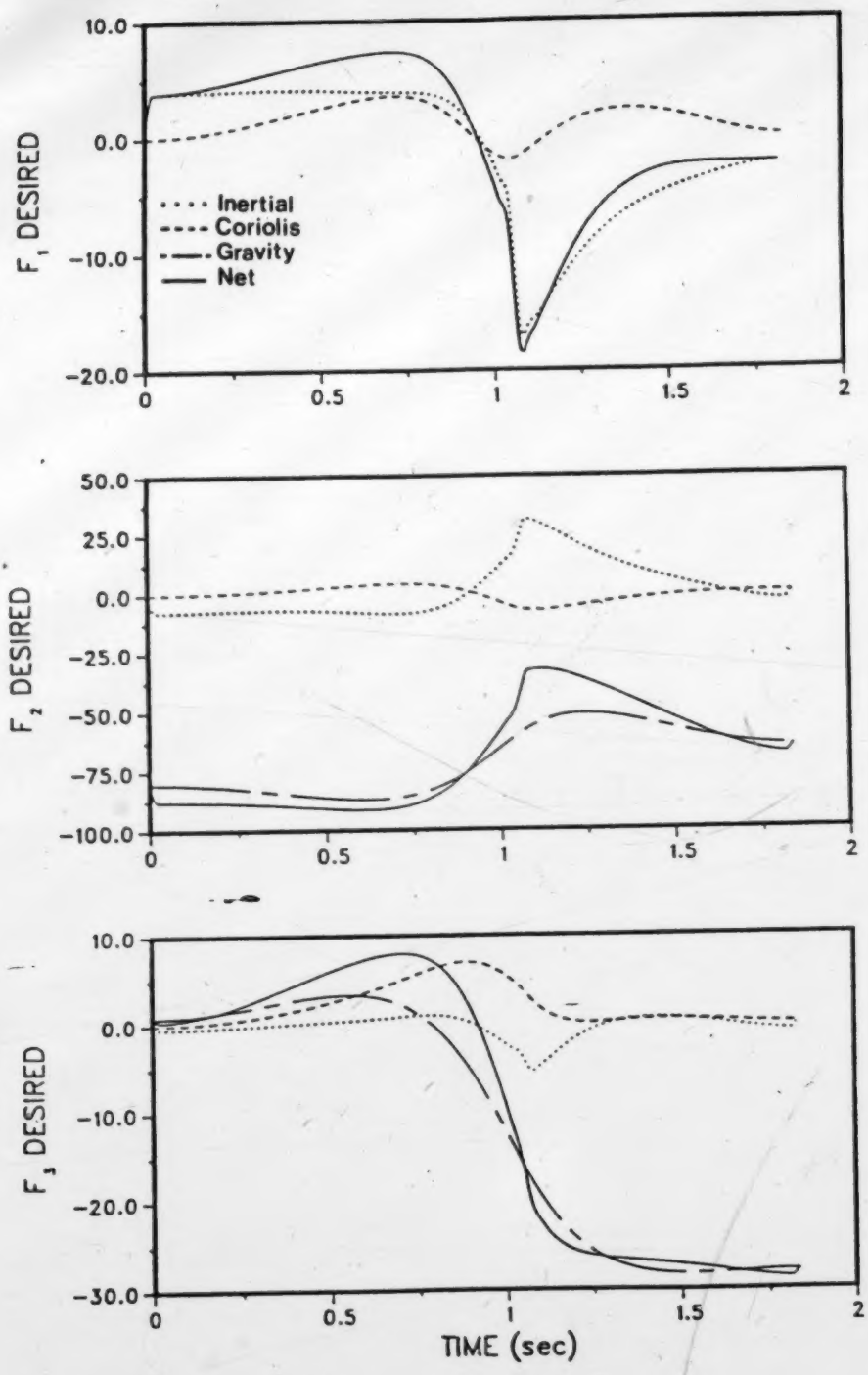


Figure 3.1 Inverse Dynamic Solution for the Free Motion Phase

will reduce the overall stability of the configuration. The PID application given in [21] did find that gravity feedforward significantly improved controller performance.

The inertial matrix $D(\theta)$ is also of considerable interest since it can be used in variable gain controllers to ensure critical damping is achieved at each point in the workspace [32]. Inertial feedforward has been proposed in numerous papers [30,31] and a review is presented in [32]. In [2] it is proposed to feedforward only the cross-coupled inertial torques at each joint (i.e. the off-diagonal terms in the $D(\theta)$ matrix. In all such cases, the task planner must provide accurate joint accelerations (second derivative). In general, the joint velocities may be obtained from the Jacobian transformation between $[\dot{x},\dot{y},\dot{z}]$ and $[\dot{\theta}_1,\dot{\theta}_2,\dot{\theta}_3]$ (see equation 2.1). However, repeated applications of the Jacobian become unwieldly and finite difference approximations [32] are often used in place of exact solutions. In either case, obtaining the second derivative accurately may become numerically too intensive to justify feedforward and a recursive inverse dynamic solution would be more easily implemented. Software for the inverse dynamics is given in Appendix 2.

For limited ranges of motion, the linearized state-space model is useful for controller design applications. Linearizing equation 3.1 about $[\theta,\dot{\theta},\ddot{\theta}]^0$ results in

$$\Delta T = \left\{ \frac{\partial D(\theta)\ddot{\theta}}{\partial \theta} + \frac{\partial C(\theta,\dot{\theta})}{\partial \theta} + \frac{\partial G(\theta)}{\partial \theta} \right\}^0 \Delta\theta + \left\{ \frac{\partial C(\theta,\dot{\theta})}{\partial \dot{\theta}} \right\}^0 \Delta\dot{\theta} + \{ D(\theta) \}^0 \Delta\ddot{\theta} \tag{3.2}$$

The linearized state space model

$$\Delta\dot{X} = A\Delta X + B\Delta u \tag{3.3}$$

is obtained by setting

$\Delta X^T = [\Delta\theta^T, \Delta\dot{\theta}^T]$.

This results in the following A and B matrices:

$$A = \begin{bmatrix} 0_{NxN} & I_{NxN} \\ -\left[D^0\right]^{-1}\left[S_D{}^0 + S_C{}^0 + S_G{}^0\right]_{NxN} & -\left[D^0\right]^{-1} S_V \end{bmatrix}_{2Nx2N}$$

$$B = \begin{bmatrix} 0_{NxN} \\ \left[D^0\right]^{-1}_{NxN} \end{bmatrix}$$

Where, the system sensitivity matrices S_D, S_C, S_V, and S_G

are defined as:

$$\frac{\partial D(\theta)\ddot{\theta}}{\partial \theta} = S_D$$

$$\frac{\partial C(\theta,\dot{\theta})}{\partial \theta} = S_C$$

$$\frac{\partial C(\theta,\dot{\theta})}{\partial \dot{\theta}} = S_V$$

$$\frac{\partial G(\theta)}{\partial \theta} = S_G$$

For single input single output control, equation 3.3 is more conveniently expressed in decoupled scalar form. The i^{th} scalar equation may be obtained by setting:

$$A_1 = -\left[D^0\right]^{-1} S_v \tag{3.4a}$$

$$A_2 = -\left[D^0\right]^{-1}\left[S_D{}^0 + S_C{}^0 + S_V{}^0\right] \tag{3.4b}$$

$$B_1 = \left[D^0\right]^{-1} \tag{3.4c}$$

which yields

$$\Delta\ddot{\theta}_i + a_{1ii}\Delta\dot{\theta}_i + a_{2ii}\Delta\theta_i = b_{1ii}\Delta T + \sum_{j\neq i}(b_{1ij}\Delta T_j - a_{1ij}\Delta\dot{\theta}_j - a_{2ij}\Delta\theta_j) \quad (3.5)$$

Equation 3.5 is useful in designing linearized statespace control. The $\Sigma(\cdot)$ term is used to decouple the state matrices. Should the control system be operating without any inverse dynamic feedforward, this term would appear as an incremental disturbance seen by the controller. For the matrix elements in A_1 and A_2 the reader may refer to the software listing in Appendix 3.

The dynamic model of the arm presented so far, does not account for any servo-motor dynamics. This creates problems when inverse dynamic solutions are used or when controller designs are based strictly on the arm model. A discussion of the servo-motor dynamics follows.

3.2 Servo-motor Dynamics

The actuators used in industrial robots are either electrical, hydraulic, or pneumatic with permanent magnet DC motors being the most common. Assuming the motor is voltage controlled and operates in the linear range, the motor may be described by the following familiar equations:

$$K_{dr}u(t) - K_m\alpha\,\dot{\theta}(t) = RI(t) \tag{3.6a}$$

$$J\,\ddot{\theta}(t) + B\,\dot{\theta}(t) = T(t) \tag{3.6b}$$

$$T(t) = \frac{1}{\alpha}K_I I(t) \tag{3.6c}$$

where:

K_{dr} - gain of voltage driver

u - control input

K_m - back emf constant

α - gear reduction ratio

θ - link angle ($1/\alpha$ X motor angle)

R - resistance of armature circuit winding

I - armature current

J - effective inertia of the assembly at the link side

B - effective viscous damping coefficient (friction)

T - generated torque at the link side

K_I - motor torque constant

This model neglects any inductance associated with the armature circuit winding. Equation 3.6b describes the link side rotation assuming the gravity center of the rotating mass lies on the axis of the shaft. In [20], a very detailed analysis of such a model and some typical data is given. Equations (3.6a) and (3.6c) may be combined so as to yield the servo-motor torque generated at the link side as a

function of the control signal u(t). This yields:

$$T_i = \frac{K_I K_{dr}}{R\alpha} \left[u(t) - \frac{K_m \alpha}{K_{dr}} \dot{\theta} \right] \qquad (3.6d)$$

3.3 Modified Manipulator Dynamics

The torque developed at the link side, T_i in equation 3.6d, represents the torque that appears in the dynamic equation describing the manipulator (equation 3.1). In addition, it can be seen from equation 3.6b, that the servo motor moment of inertia and the viscous damping coefficient must both be added to the link side of equation 3.1. The servo-motor moment of inertia is constant for the entire range motion while the viscous damping may, as a first approximation, be considered to be pure *dynamic friction*. This approximation yields a constant for the viscous damping coefficient (B in equation 3.6b), though in general it may depend on both θ and $\dot{\theta}$. In obtaining the incremental linearized model then, it can be seen that the servo inertia should be added to the diagonal terms in $D(\theta)$ and that the viscous damping term is added to the coriolis torque vector prior to linearization (equation 3.1) This yields a modified linearized model given by equation 3.4d.

$$\Delta\ddot{\theta}_i + \{a^*_{1ii} + B_i + \frac{b_{1ii}K_IK_m}{R}\}\Delta\dot{\theta}_i + a^*_{2ii}\Delta\theta_i = b^*_{1ii}\frac{K_IK_{dr}}{R\alpha}u(t) + \sum_{j\neq i}(b^*_{1j}\Delta T_j - a^*_{1ij}\Delta\dot{\theta}_j - a^*_{2ij}\Delta\theta_j) \quad (3.4d)$$

where, the superscript * is used to indicate that the diagonal terms in $D(\theta)$ have been increased by the inertia of the servo motors; and B_i is included to reflect any viscous damping or friction. Taking the Laplace transform of equation (3.4b) and ignoring the disturbance term, yields:

$$\frac{\Delta\theta_i}{\Delta u_i} = \frac{1}{a_2s^2 + a_1s + a_0} \quad (3.7)$$

where the joint subscript has been dropped for notational simplicity. Each of the modified coefficients are defined below:

$$a_2 = \frac{R\alpha}{b^*_{1ii}K_IK_{dr}}$$

$$a_1 = \{a^*_{1ii} + B_i + b_i\frac{K_IK_m}{R}\}\frac{R\alpha}{b_{1ii}K_IK_{dr}}$$

$$a_0 = \{a^*_{2ii}\}\frac{R\alpha}{b_{1ii}K_IK_{dr}}$$

From the range of workspace and loads, each of the constants appearing in (3.7) are bounded between a global minimum a_{im} and a global maximum a_{iM}.

$$\begin{aligned} a_2 &\in \{a_{2m}, a_{2M}\} \\ a_1 &\in \{a_{1m}, a_{1M}\} \\ a_0 &\in \{a_{0m}, a_{0M}\} \end{aligned} \tag{3.8}$$

Each interval in 3.8 is assumed to be known and is essential to the design procedure.

Equation 3.7 represents a linearized transfer function that explicitly includes both the servo-motor dynamics and the coupled link dynamics. In using this transfer function for design purposes, no assumptions need to be made about the type of motion under control. The bounds used in equation 3.8 may be evaluated for specific motions by utilizing the inverse dynamic software given in Appendix 2. To this authors knowledge, no general analytic method is available to generate the intervals for the entire range of workspace and motions. However, by utilizing the symbolically generated linearized equations, a reasonable estimate of these parameters may be obtained.

4. Fixed Gain Conventional Servo Design

4.1 Introduction

To date, industry has primarily relied on conventional single-input-single-output (SISO) fixed gain proportional derivative control (PD). Such control applies over the entire workspace and range of loads (robustness), and is easily implemented in real-time.

Although PD control may work well for free motions, it is typically inadequate for guarded or compliant motions. These tasks generally have extremely tight tolerances. One obvious method to improve accuracy is to include an integral mode. Although this has been recognized by a number of authors [1,9,20], a simple methodology for obtaining the PID controller gains has yet to be proposed.

To evaluate the performance of conventional controllers, the positional PD servo design presented in [2] is briefly reviewed. This design is based on the structural resonant frequency of the robot and the desired closed loop damping ratio. The design includes effective inertias (link + servo motor) in developing transfer functions and considers all other dynamic forces to be disturbances. It should be noted that the structural resonant frequency of a particular robot is an experimentally determined parameter. This parameter results from non-rigid links, and shaft torsional resistance. Neither of these phenomena is dealt with in the dynamic model presented in chapter 3.

An extension of the the design method proposed in [2] is then developed so as to include gravity forces in the transfer function describing an individual joint. Although this model still treats coriolis and cross coupled inertial torques as disturbances, the transfer function itself is an accurate description of both the arm dynamics and the servo parameters. The structure of the resulting controller is identical to that given in [2]. The essential difference occurs in the design methodology which yields different controller settings.

For the guarded phase a special split-P-mode controller which utilizes two separate proportional control modes is proposed. A design methodology for obtaining the controller gains is also presented. This controller utilizes a PI controller with the previous PD design. The design is primarily based on classical root-locus techniques [24].

For the compliant phase, a force feedback loop is cascaded with the split-P-mode controller. This results in a controller that incorporates both positional and force feedback as opposed to hybrid designs that drop the positional feedback along the force controlled axes. Once more the design procedure incorporates root-locus techniques.

For all phases, the designs yield control gains that may be used over the entire range of motion and loads. The designs are verified, through simulation studies, for the task described in Chapter 2.

4.2 Servo Design Technique Based on a Simplified Model

Using the servo-motor equations presented in chapter 3 (equation 3.5) the transfer function describing the motor dynamics is determined to be:

$$\frac{\theta(s)}{u(s)} = \frac{1}{s(a_2 s + a_1)} \qquad (4.1)$$

where

$$a_1 = (B + \frac{K_m K_I}{R}) \frac{R\alpha}{K_I K_{dr}} , \qquad a_2 = \frac{JR\alpha}{K_I K_{dr}}$$

This is an integral-type transfer function with a time constant:

$$a_1/a_2 = JR/(BR + K_m K_I)$$

Equation (4.1) is the familiar servo transfer function used in classical servo control design.

The effective inertia J_{eff} [1] of the assembly can be written as:

$$J_{eff} = D(\theta) + I\alpha^2 \qquad (4.2)$$

where $D(\theta)$ is inertia of the link, obtained from the dynamic equations of motion which characterize the robot, and $\alpha^2 I$ is the inertia of the motor (constant) increased by the gear ratio α (also constant). For simulation purposes, the inertias of the three servo motors increased by their respective gear rations are taken (SI units) as $\alpha^2 I_1 = 2.0$, $\alpha^2 I_2 = 3.0$, and $\alpha^2 I_3 = 1.0$. Furthermore, it is assumed that:

$$\frac{K_I K_{dr}}{R\alpha} = 1.0$$

Should this identity not be true, an additional gain block could be added to the overall system. Since $D(\theta)$ varies with the arm configuration, the coefficient a_2 varies as well. In addition, the viscous damping coefficient B may only be specified within a rather limited accuracy, as it depends on the condition and maintenance of the robot. This in turn affects the coefficient a_1. Hence the lower and upper bounds on a_1 and a_2 may be expressed as:

$$a_1 \in \{a_{1m}, a_{1M}\}, \quad a_2 \in \{a_{2m}, a_{2M}\} \tag{4.3}$$

These intervals are assumed known and are essential to the design procedure.

A conventional PD servo control system is shown in Figure 4.1 where θ_r denotes the reference link angle (provided by the trajectory planner), and k_p, k_d are the gains of the P and D mode respectively. The derivative action does not act directly on the reference so as to avoid overshoots. The closed-loop transfer function of the system is given by:

$$\frac{\theta(s)}{\theta_r(s)} = \frac{\frac{K_p}{a_2}}{s^2 + \frac{a_1 + K_d}{a_2} s + \frac{K_p}{a_2}} \tag{4.4}$$

from which the undamped natural frequency and damping ratio are determined to be:

$$\omega_n = \sqrt{\frac{K_p}{a_2}} \quad , \zeta = \frac{a_1 + K_d}{2} \frac{1}{\sqrt{a_2 k_p}} \tag{4.5}$$

In Paul [2], two fundamental design criterion are proposed:

$$\begin{aligned} &\text{(i)} \quad \omega_n \leq \tfrac{1}{2} \omega_{rST} \\ &\text{(ii)} \quad \zeta \geq 1 \end{aligned} \tag{4.6}$$

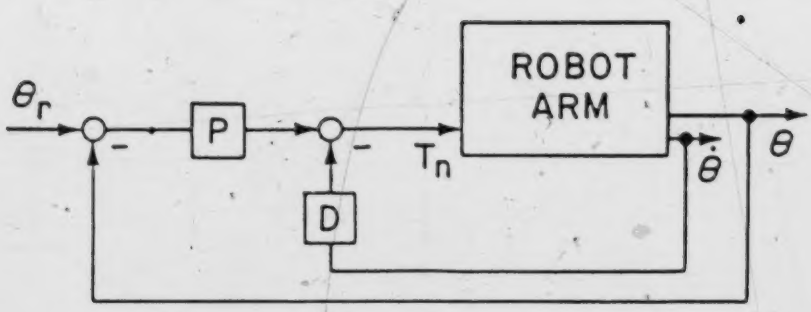


Figure 4.1 Conventional PD Servo Control System

where ω_{rST} is the structural resonant frequency which in turn is a function of the joints stiffness K and the effective inertia J

$$\omega_{rST} = \sqrt{\frac{K}{J_{eff}}} \tag{4.7}$$

Since J_{eff} depends on the load, so does ω_{rST} Thus:

$$\omega_r \in \{\omega_{rm}, \omega_{rM}\} \tag{4.8}$$

For fixed gain control, the inequalities in (4.6) must be satisfied over the entire workspace and range of loads thus, the following conditions (from 4.7) must hold:

$$K_p \leq \frac{1}{4} a_{2m}\omega_{rm}^2, \qquad K_d \leq 2\sqrt{a_{2M}K_p - a_{1m}} \tag{4.9}$$

This constitutes the essential result in Paul [2].

The servo transfer function (4.1) assumes that the gravity centers of the rotating masses are located on the axes of the shafts, which in turn implies that the gravity torques are zero. Since this not usually the case, gravity

torques are often eliminated by precalculating feedforward compensation [2,20]. This is not straight forward since feedforward gains depend on arm configuration and load. The whole control system becomes complicated and hence is rarely used in industry. If the gravitational force is not compensated for, the conditions in (4.6) must be satisfied with certain safety margins so as to prevent oscillatory responses.

4.3 Free motion PD Control Utilizing a Full Dynamic Model

To obtain controller gains which account for both the servo-motor and the arm-motion, the transfer function established in equation (3.6) is utilized. For the PD controller shown in Figure 4.1, and the open-loop transfer function in 3.6, the closed-loop transfer function is obtained.

$$\frac{\Delta\theta(s)}{\Delta\theta_r(s)} = \frac{K_p}{a_0 + K_p} \quad \frac{\frac{a_0 + K_p}{a_2}}{s^2 + \frac{a_1 + K_d}{a_2} s + \frac{a_0 + K_p}{a_2}} \tag{4.10}$$

from which ω_n and ζ are determined.

$$\omega_n = \sqrt{\frac{a_0 + K_p}{a_2}}, \quad \zeta = \frac{a_1 + K_d}{2} \quad \frac{1}{\sqrt{a_2 (a_0 + K_p)}} \tag{4.11}$$

Equation (4.10) clearly indicates the presence of a steady state incremental positional error of magnitude

$$\Delta e_{ss} = \Delta\theta_r \, K_p/(a_0+K_p)$$

From equation (4.11) and the constraints in (4.6), the controller settings are obtained (using equalities) as:

$$K_p = \frac{1}{4} a_{2m}\omega_{rm}^2 - a_{0M}, \quad K_d = 2\sqrt{a_{2M}(a_{0M} + K_p)} - a_{1m} \tag{4.12}$$

Since the coefficients a_0, a_1, and a_2 vary with arm configuration and load (time) so do the closed-loop poles. Denote these poles as $-s_1$, $-s_2$ with $s_2 \geq s_1$. The bounds of the intervals $\{s_{1m}, s_{1M}\}$, $\{s_{2m}, s_{2M}\}$, within which s_1, s_2 are located can be calculated as

$$s_{1m} = \frac{1}{2}\left[\frac{a_{1M} + K_d}{a_{2m}} - \sqrt{\left(\frac{a_{1M} + K_d}{a_{2m}}\right)^2 - 4\,\frac{a_{0m} + K_p}{a_{2m}}}\right]$$

$$s_{1M} = s_{2m} = \frac{1}{2}\,\frac{a_{1m} + K_d}{a_{2M}} \tag{4.13}$$

The expression for s_{2M} may be obtained from s_{1m} by using a positive square root. Knowing the range of poles, one may

estimate the settling time of the system. If s_1, s_2 are close to $s_{1M} = s_{2M}$ then the step response truly represents a second order system. If, however, s_1 is close to s_{1m} (and s_2 to s_{2M}) then the first order approximation seems more appropriate. In this case, the approximate settling time is:

$$t_{s(PD)} = \max\{ 6 \frac{1}{s_{1M}} , 3 \frac{1}{s_{1m}} \} \qquad (4.14)$$

4.4 Guarded Motion PID Control

Since the PD control scheme discussed in the previous section results in steady state errors it is not suitable for guarded or compliant motions. In order to overcome this problem an integral (I) mode is incorporated into the control algorithm. From the many possible PID schemes, the one shown in Figure 4.2. is chosen for design simplicity. It should be noted that feedforward compensation may also be used to increase accuracy (as was done in Paul [2]). The advantage of using integral action, as apposed to feedforward techniques, lies in the fact that steady state errors are guaranteed to go to zero regardless of parameter inaccuracies and modeling errors. Integral control does however take longer to settle and does not yield zero error while tracking a dynamic reference.

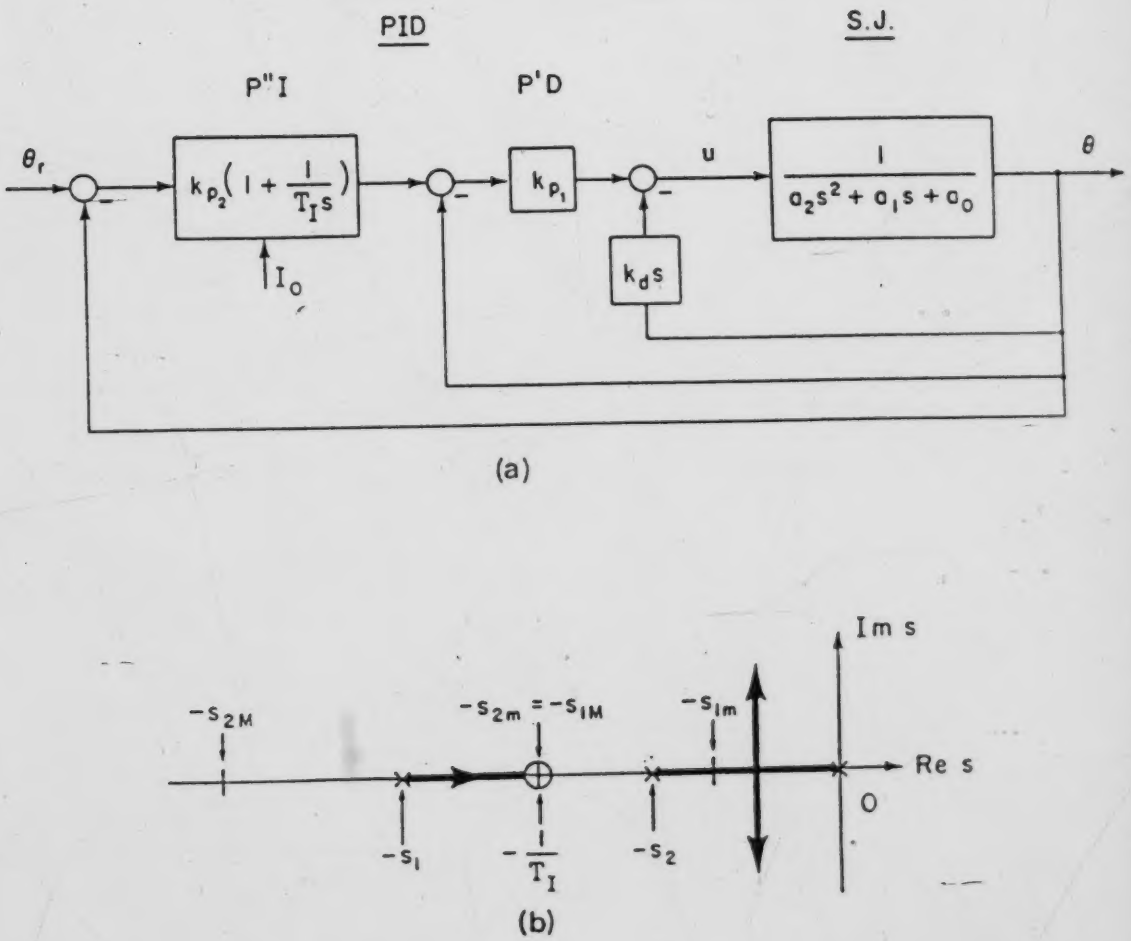


Figure 4.2 Guarded Motion PID Control Scheme: a) Block Diagram b) Root-locus

In Figure 4.2a, a PI control signal is added to the PD signal. This can be described as a "split-P-mode" controller, since two proportional blocks are employed. The root-locus plot of such a system is shown in Figure 4.2b for the case when

$$T_I = \frac{1}{s_{1M}} \tag{4.15}$$

where s_{1M} is defined in (4.13). While other choices of T_I are possible (for example, $T_I \geq 1/s_{1M}$, would leave the overall root-locus unchanged) use of (4.15) is recommended since the strong integral action yields excellent accuracy, and yet at the same time is not so strong as to interfere with the dominant branches originating at zero and $-s_1$. Note that moving $-1/T_I$ to the right from $-s_{1M}$ may cause overshoots.

The characteristic equation of the closed loop system is:

$$1 + K_{p2}\left(1 + \frac{1}{T_I s}\right) \frac{K_{p1}}{a_2} \frac{1}{(s + s_1)(s + s_2)} = 0 \tag{4.16}$$

From (4.13), it is clear that the breakaway point of the two dominant branches is located to the left of $-s_{1m}/2$ (fig. 7b). It was pointed out in the previous section, that the

rightmost location corresponds to the case when $s_1 = s_{1m}$ and $s_2 = s_{2M}$. Therefore, to avoid underdamping or large overdamping, it is convenient to choose a closed-loop pole at $-s_{1m}/2$. Application of the magnitude condition to (4.16) together with (4.15) yields

$$K_{p2} = \frac{a_{2m}}{K_{p1}} \left[\frac{s_{1m}}{2} \right]^2 \frac{s_{2M} - s_{1m/2}}{s_{1M} - s_{1m/2}} \qquad (4.17)$$

The presence of a_{2m} in equation (4.13) reflects the bounds for which s_{1m} and s_{2M} are evaluated. The formulae (4.12), (4.15), and (4.17) specify the settings of the "split-P-mode" controller.

Since the two dominant closed-loop poles (located near $-s_{1m}/2$) determine the dynamics of the system, the settling time can be roughly estimated as (neglecting the third branch):

$$t_{s(PID)} = 6 \frac{1}{s_{1m\ /2}} = 12 \frac{1}{s_{1m}} \qquad (4.18)$$

Comparing (4.14) with (4.18) it is seen that the settling time with the PID controller is approximately four times that obtained with PD controller. This characteristic of PID

control makes it a less desirable choice for the free motion phase where speed is likely to be of greater importance than zero steady state error. The PID control law may be explicitly written as:

$$u(s) = K_{P2}K_{P1}\left(1 + \frac{1}{T_I s}\right)\theta_r(s) - \left[K_{P2}(K_{P1} + 1) + K_{P2}K_{P1}\frac{1}{T_I s} + K_d s\right]\theta(s) \qquad (4.19)$$

In practical applications, a smooth transfer of the control signal u(t) is required when switching from PD to PID. This "bumpless" transfer may be obtained by setting the appropriate initial condition at the integral mode. Specifically, let t_0 be the moment when switching occurs. Denote the output of the integral mode as:

$$\frac{K_P}{T_I} = \int(\theta_r(\tau) - \theta(\tau))\, d\tau + I_0$$

Then, the initial condition I_0 (Figure 4.2a) should be set to:

$$I_0 = \frac{T_I}{K_{p2}} [\theta_r(t_0) - K_{P2}(\theta_r(t_0) - \theta(t_0))] \qquad (4.20)$$

The concept of a "split-P-mode" controller can of course be applied to the designs of Paul or Luh [2,3].

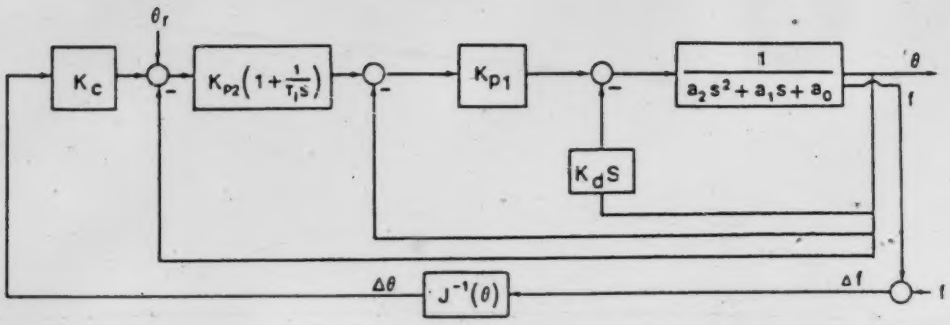
4.5 Compliant Motion Cascade Control

Once contact with the target surface has been made, the manipulator enters the compliant motion phase. During this phase it is necessary to maintain a force balance between each of the force sensors shown in Figure 2.3. For this purpose, a force feedback loop is added to the previously designed guarded motion controller as shown in Figure 4.3. Since this scheme retains the position and velocity feedback, it is ideally suited to applications requiring a range of compliant motion.

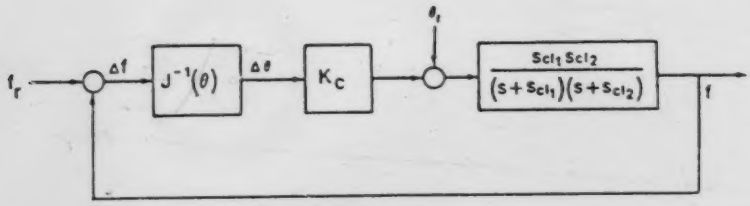
From Figure 4.3a, define

$$\frac{\theta(s)}{\theta_r(s)} \Bigg|_{K_c\,\Delta(\theta) = 0} = \frac{\theta(s)}{K_c \Delta(\theta)} \Bigg|_{\theta_r(s) = 0} = G(s) \qquad (4.28)$$

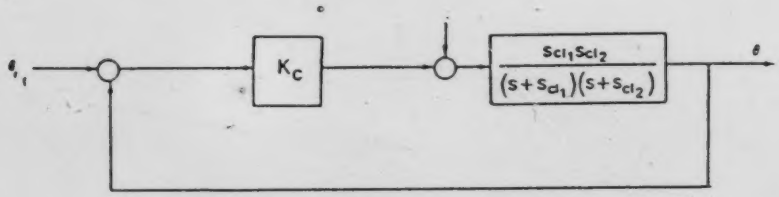
It can be shown that with $T_I = 1/s_{1M}$, as in (21), and $s_1 \simeq s_{1M}$, G(s) reduces to a second order transfer function with



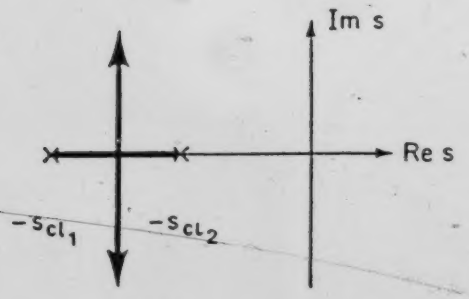
(a)



(b)



(c)



(d)

Figure 4.3 Compliant Motion Control Scheme: a) Block Diagram, b,c) Simplified block diagrams, d) Root-locus plot.

two real poles and unity gain. That is,

$$G(s) = \frac{s_{cl_1}\, s_{cl_2}}{(s + s_{cl_1})(s + s_{cl_2})} \tag{4.22}$$

For this transfer function, the block diagram shown in Figure 4.3a can be reduced to that shown in figure 4.3b. A further simplification can be performed by noting

$$\begin{aligned} \Delta\theta &= J^{-1}(\theta)\ (\ f_r - f\ ) \\ &= \theta_r - \theta \end{aligned} \tag{4.23}$$

This modification is shown in Figure 4.3c. The root-locus for this system is shown in figure 4.3d for $0 \leq K_c \leq \infty$. A procedure for choosing K_c is discussed next.

In the design of the guarded motion controller, both closed loop poles were chosen to be near $-s_{1m}/2$. Approximating $s_{cl}=-s_{cl1}=-s_{cl2}$ one can further simplify the open-loop transfer function as

$$G(s) = [s_{cl}/(s + s_{cl})]^2$$

which yields a closed-loop characteristic equation:

$$s^2 + 2s_{c1}s + (k_c + 1)s_{c1}^2 = 0 \qquad (4.24)$$

This equation will have complex roots for $K_c \geq 0$. Let ζ_c, the damping ratio of these roots, be a design parameter. From (4.24), we get

$$K_c = \frac{1}{\zeta_c^2} - 1 , \quad \zeta_c \leq 1 \qquad (4.25)$$

To avoid excessive overshoots ζ_c should be chosen between 0.45 and 0.70 which results in K_c between 1 and 4. From Figure 4.3d, it is clear that overshoots are unlikely except for the critical situation involving the limiting values of coefficients. That is, ζ will be only be less than 1.0 for $s_{c11} = s_{c12}$. For K_c between 1 and 4 the response should not be overly different than the critically damped response. Hence, from (4.24), taking into account that s_{c1} is near $s_{1m}/2$, the settling time may be estimated as

$$t_{settle,cascade} = 6 \frac{1}{s_{1m/2}} = 12 \frac{1}{s_{1m}} \qquad (4.26)$$

This is the same value as in (4.18)

4.6 Simulation Results

To obtain simulation results, the dynamic equations of motion (Appendix 1) were solved utilizing forth-order Runge-Kutta techniques. For this purpose the equations were first translated using the continuous simulation language ACSL [26]. The ACSL source program is given in Appendix 3. For design purposes, the bounds used on the intervals given in equation 3.6 were taken as shown in Table 2.

TABLE 4.2

Link	a_{0m}	a_{0M}	a_{1m}	a_{1M}	a_{2m}	a_{2M}
1	-5.	2.0	0.0	20.0	3.0	5.0
2	-4	+5.	-90.	60.0	4.0	7.0
3	-1	+1	-50.	30.0	1.0	2.0

The minimum structural resonant frequencies of the links are assumed to be :

f_{rST1} = 4.0 Hz, f_{rST2} = 6.0 Hz, f_{rST3} = 10.0 Hz

which yield the controller settings in equations 4.12, 4.15 and 4.18. Similarly, for the compliant motion controller operating with ζ_c=0.7, the control gains in (4.25) may also

be obtained.

Shown in Figure 4.4, (solid lines) are the desired joint obtained from equation (2.2) and (2.3). The dashed line indicates the values of the joint variables obtained using the PD free-motion control. It should be noted that at the end of the free motion (1.87 seconds) the robot is not in steady state. At this point, the guarded motion controller assumes control. Figure 4.5 shows the cartesian errors associated with this motion. The dotted line indicates contact with the rigid surface at which time (1.0 seconds) the compliant motion controller assumes control. At t=4 seconds, y_{ref} decelerates (see figure 2.7). This causes the arm to drop slightly increasing the errors in Z and Y. At t=8.0 seconds, all references are constant and the arm is allowed to reach steady state (zero error). By comparing the dotted line and the solid line (Figure 4.5) it can be seen that the addition of force sensors clearly improves the tracking errors. This is especially important whenever the arm has to work against gravity.

From Figure 4.5, it should be noted that all the errors are within about 1.0 cm at the instant the arm makes contact. Although the PUMA arm is capable of repeatabilities to 1/5000 of an inch, its accuracy is not likely much greater than a few centimeters. That is, the controller proposed here is likely operating within the detectable accuracy of the inverse kinematic solution considering errors in measuring θ and general inaccuracy in the link

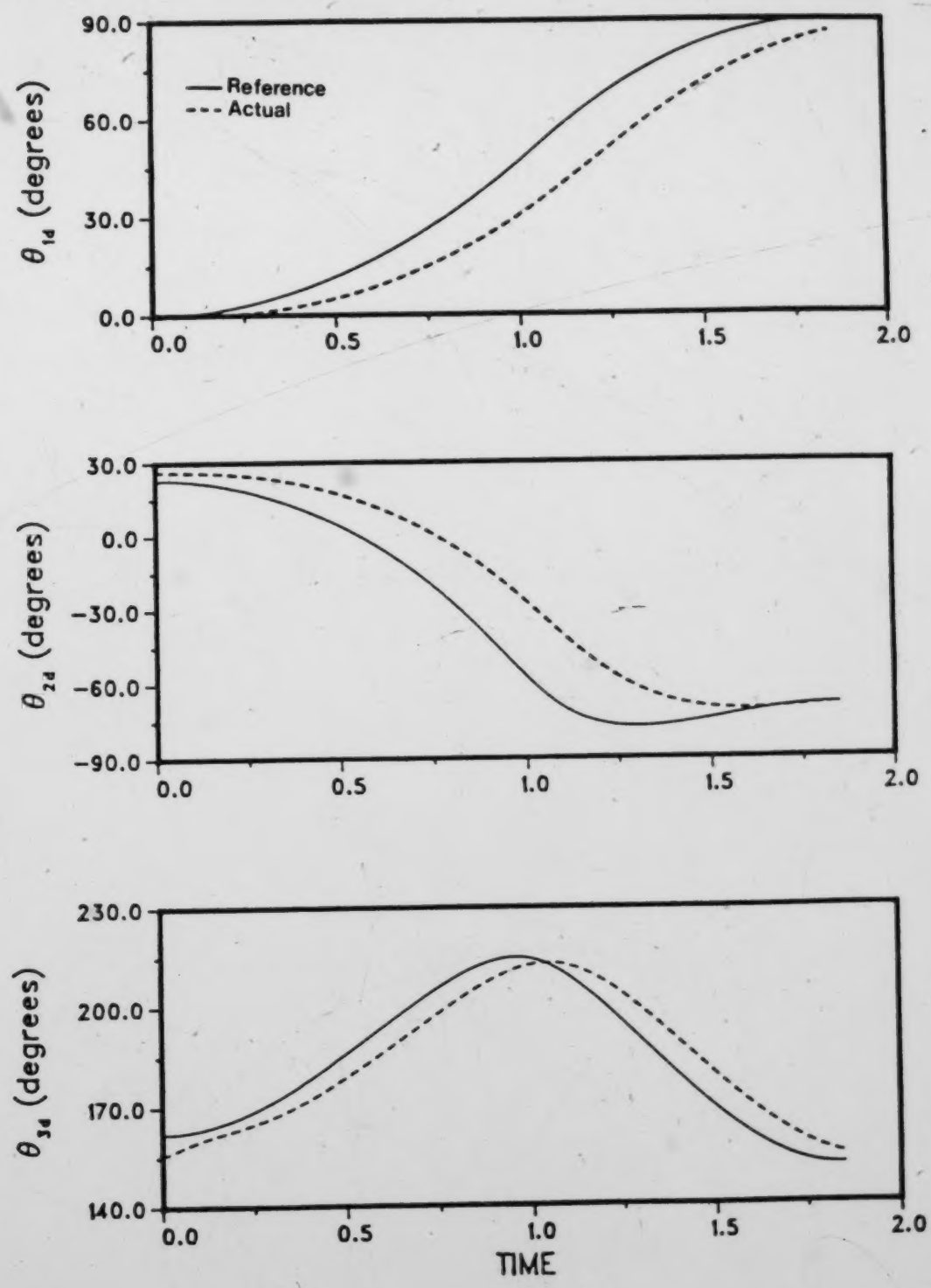


Figure 4.4 Free Motion Joint Reference Tracking

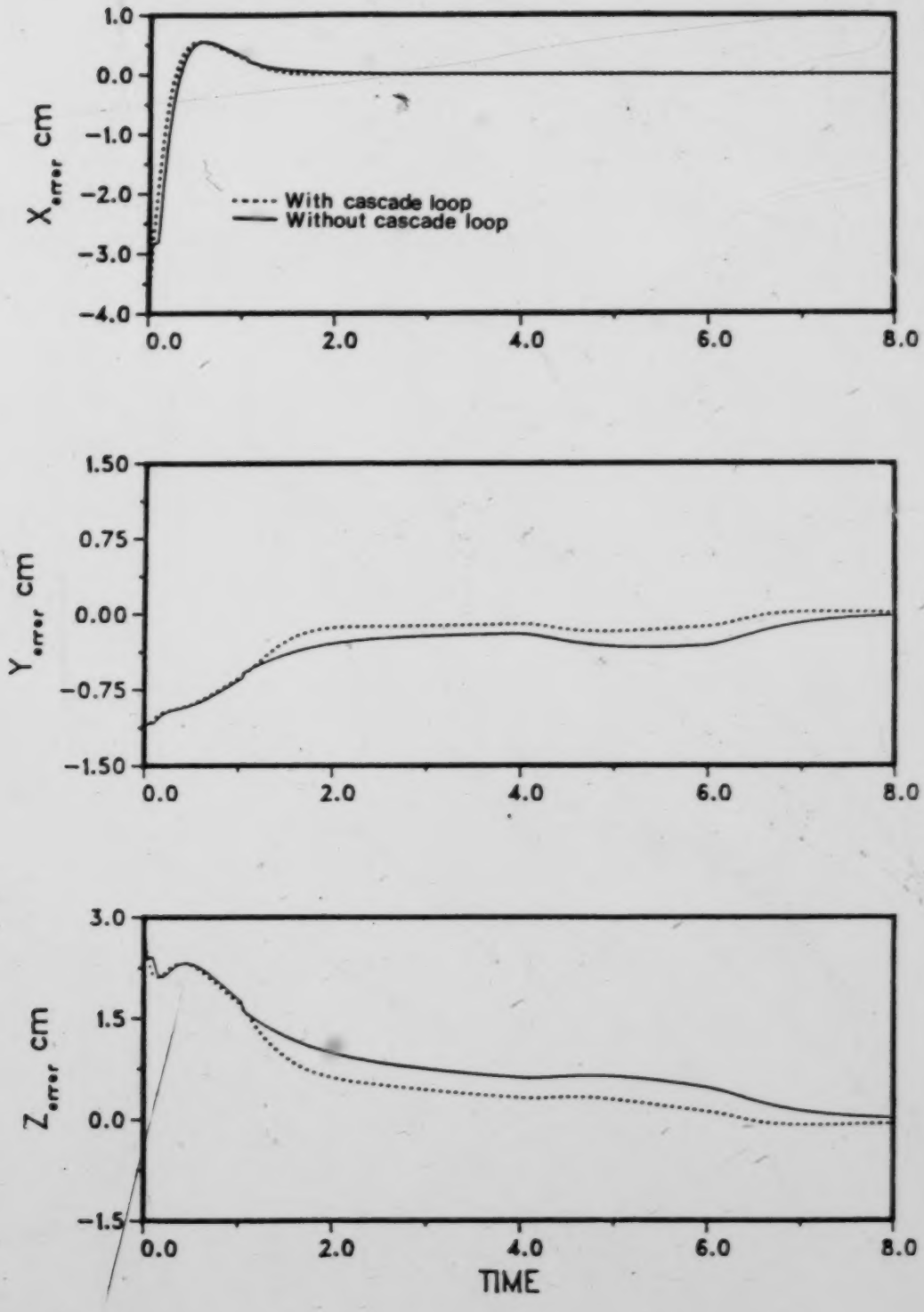


Figure 4.5 Guarded and Compliant Motion Cartesian Errors

parameters. A complete kinematic error analysis for the PUMA is given in [27].

5. State Space Controller Design Enhancements

5.1 Introduction

In this Chapter, the basic controller designs presented in Chapter 4 are enhanced. For the free motion phase, a state variable reference approach is used in conjunction with nonlinear feedforward. Control enhancements for the guarded, and compliant phases are proposed based on the linearized state matrices developed in Chapter 3. These control strategies are only implemented for the guarded and compliant motions since the linearized model is valid only for limited ranges of motion.

For the free-motion phase, the state variables are chosen as:

$$X^T = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3] \qquad (5.1)$$

To insure steady state accuracy during the guarded and compliant phases, the state variable assignment is augmented by including the integral of each positional error as a new state variable.

$$X^T = [\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, q_1, q_2, q_3] \qquad (5.2)$$

where $q_i = \int \theta_i - \theta_{ir}\, dt$ For each phase, the control law is chosen as:

$$u(t) = -K[\, X(t) - X_d(t)] \qquad (5.3)$$

where X_d is the desired state (q_d would be set to zero).

5.2 Free Motion State-space Design

Comparing the classical PD design presented in Chapter 4 to the state feedback control law given in equation 5.3, it becomes obvious that for a second order system the two control laws are equivalent with the exception that in the state-feedback design the derivative action acts on the derivative of the error rather then the derivative of the output. Originally, in the classical design the derivative action was applied only to the output so as to prevent overshoots. In general, a state-feedback system may have real eigenvalues and still exhibit overshoots. Modifications, which limit the amount of control action produced by the reference velocities, may be required here. This could be accomplished by reducing the desired velocities to be a percentage α of the true velocity references. That is to say, given θ_{ir} (reference position), the reference velocity would be given as:

$$\theta_{1r} = a_i \frac{d}{dt} \theta_{ir}$$

Note that with a set to zero the derivative action acts on the output alone and that with a set to 1.0 the derivative action acts on the error alone. In general, it can be shown for a second order response, setting a equal to the reciprocal of the smaller time constant is a sufficient condition to insure zero overshoot, although this condition will result in an extremely conservative design. The block diagram for free motion state-controller is shown in Figure 5.1.

5.3 Guarded and Compliant Motion State-space Design

For the final two phases, the linearized state-space model may be utilized. This model may be assumed constant over the limited range of motion associated with the final two phases. From equations 3.5 and 4.1 the following decoupled state space PID control law is formulated:

$$\Delta T_i = k_{Pi}(\Delta\theta_{ri} - \Delta\theta_i) + k_{Ii} \int (\Delta\theta_{ri} - \Delta\theta_i)dt + u_{0i} + k_{Di}(\Delta\theta_{ri} - \Delta\theta_i) - \frac{1}{b_{ii}} \sum_{j \neq i} (\cdot) \tag{5.4}$$

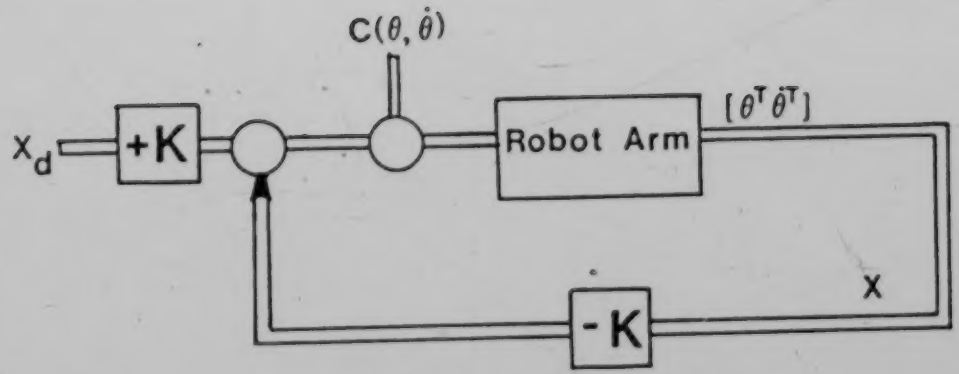


Figure 5.1 Free Motion State Space Control

Where, $\Sigma(\cdot)$ is defined in equation (3.5). This control law is shown in block diagram form in Figure 5.2. Use of this control law decouples the link equations such that the closed loop equation describing θ_i does not depend on θ_j. The term u_{0i} is included to provide bumpless transfer between the free and guarded motions. With this term set to zero the closed-loop equation becomes:

$$\Delta\dddot{\theta}_i + (a_{1ii} + b_{ii}k_{\Delta i})\Delta\ddot{\theta}_i + (a_{2ii} + b_{ii}k_{Pi})\Delta\dot{\theta}_i + b_{ii}k_{Ii}\Delta\theta_i = b_{ii}(k_{di}\Delta\ddot{\theta}_{ri} + k_{Pi}\Delta\dot{\theta}_{ri} + k_{Ii}\Delta\theta_{ri}) \tag{5.5}$$

The corresponding transfer function is given in equation (5.6) shown below:

$$\frac{\Delta\theta_i}{\Delta\theta_{ri}} = \frac{b_{ii}k_{Di}(s^2 + \frac{k_{Pi}}{k_{Di}}s + \frac{k_{Ii}}{k_{Di}})}{s^3 + (a_{1ii} + b_{ii}k_{Di})s^2 + (a_{2ii} + b_{ii}k_{Pi})s + b_{ii}k_{Ii}}$$

The three closed-loop poles in (5.6) may be arbitrarily located by suitable choices of k_{Pi}, k_{Ii}, and k_{Di}. Combining all the scalar equations in (5.4) yields the matrix gains K_P, K_I, and K_D. The the three closed-loop poles may be chosen based on the structural resonant frequency of the arm. So as to approximate a second order critically damped

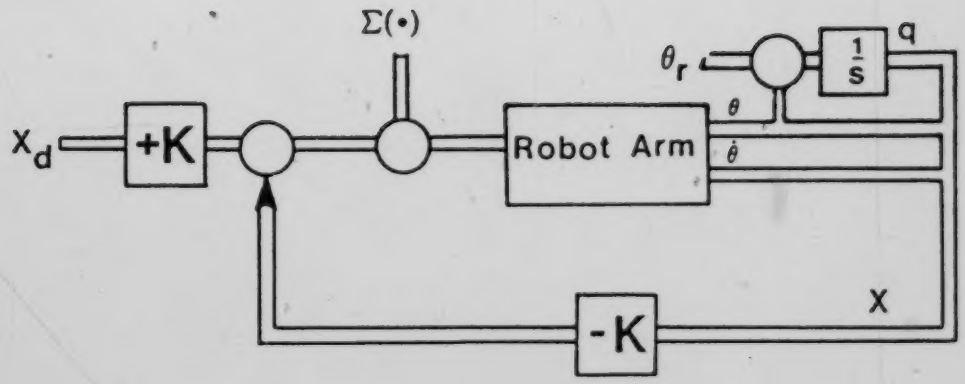


Figure 5.2 Guarded and Compliant Motiuon State Space Control

system, it is recommended that two dominant poles be chosen so as to place the the closed loop natural frequency of oscillation at a percentage ρ of the link structural resonant frequency. The third pole may then be set to be significantly greater (2 - 4 times) and thus move it further to the left in the s-plane. Denoting p_{ij} as a closed-loop pole location, where the subscript "ij" denotes the link number and pole number respectively, yields:

$$p_{i1} = p_{i2} = \rho\omega_{rSTi}$$

$$2.* p_{i1} \leq p_{i3} \leq 4.* p_{i1}$$

Note that ρ is constrained to be less than 0.5 so as to insure damping of structural resonant frequencies (equation 4.6).

Choice of k_{Fi} may be made through well known root-locus techniques. To do so, the characteristic equation for the cascaded compliant controller is first determined to be:

$$s^3 + (a_{1ii} + b_{ii}k_{Di})s^2 + (a_{2ii} + b_{ii}k_{Pi})s + b_{ii}k_{Ii} + k_{Fi}\, b_{ii}k_{Di}\left(s^2 + \frac{k_{Pi}}{k_{Di}}\, s + \frac{k_{Ii}}{k_{Pi}}\right) = 0.0 \qquad (5.7)$$

The root-locus plot of (5.7) for $0 \leq k_{Fi} \leq \infty$ is then obtained (Figure 5.3) and k_{Fi} determined based on a desired damping ratio. Since the state-equations are decoupled by the guarded motion controller, the resulting state feedback gain matrix K_F is diagonal.

5.4 Simulation Results

The improved performance of the free-motion state-feedback law given in 5.3, when compared to the classical design, is particularly evident if the speed at which the free motion is operating at is increased. This may be accomplished by increasing ϕ, shown in Figure 2.5, by a factor of 3. Keeping the control gains established in 4.12, tracking performance is shown in Figure 5.4 for $0.0 \leq \alpha \leq 1.0$. For these simulations, coriolis feedforward does not significantly improve performance.

At the start of the guarded motion phase, the operating point $\{T^0, \theta^0, \dot{\theta}^0, \ddot{\theta}^0\}$ is recorded and the resulting state matrices are determined to be:

$$A_1 = \begin{bmatrix} -0.0369900 & -0.041880 & 0.0076341 \\ 0.0059800 & -0.029946 & -0.015525 \\ -0.030159 & -0.082989 & 0.010353 \end{bmatrix}$$

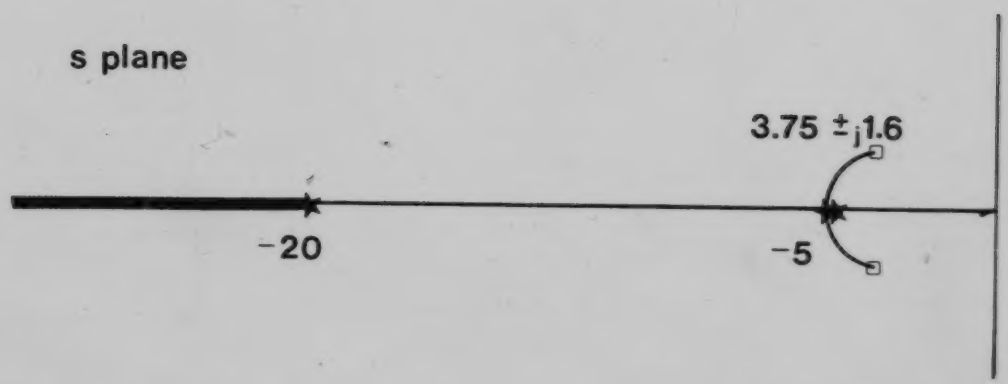


Figure 5.3 Root-locus for Cascade Control

$$A_2 = \begin{bmatrix} 0.0 & 5.1000 & -0.054531 \\ 0.0 & 12.802 & 0.70375 \\ 0.0 & -3.6287 & 1.8322 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0.24488 & 0.04029 & -0.012273 \\ 0.040219 & 0.16122 & -0.10094 \\ -0.012273 & -0.10094 & 0.94801 \end{bmatrix}$$

To locate the three closed-loop poles, ρ is first chosen to be .25 with the third pole taken at twice this value (as, before the structural resonant frequencies are taken to be 4.0, 6.0, and 10.0, respectively for links 1, 2, and 3). This results in the tracking errors shown in Figure 5.8. For ρ taken as 0.2 with the third pole located at three times this value the tracking errors are shown in Figure 5.9. Finally, two poles are arbitrarily taken at 5.0 with the third pole located at four times this value (Figure 5.10). Once more, for all cases, the inverse dynamic feedforward terms do not significantly improve performance. In addition, it can be seen that the cascaded compliant loop does not add to the overall performance since the guarded motion state-controller has essentially produced zero error from the contact point on. In practical applications,

however is not addvisable to drop this loop since the force measurement is a significantly more accurate measurment of the end effector's position relative to the slot. Furthermore, this feedback loop would correct for any misalignment of the slot (i.g. the arm might be in the correct location while the slot maybe incorrectly located.)

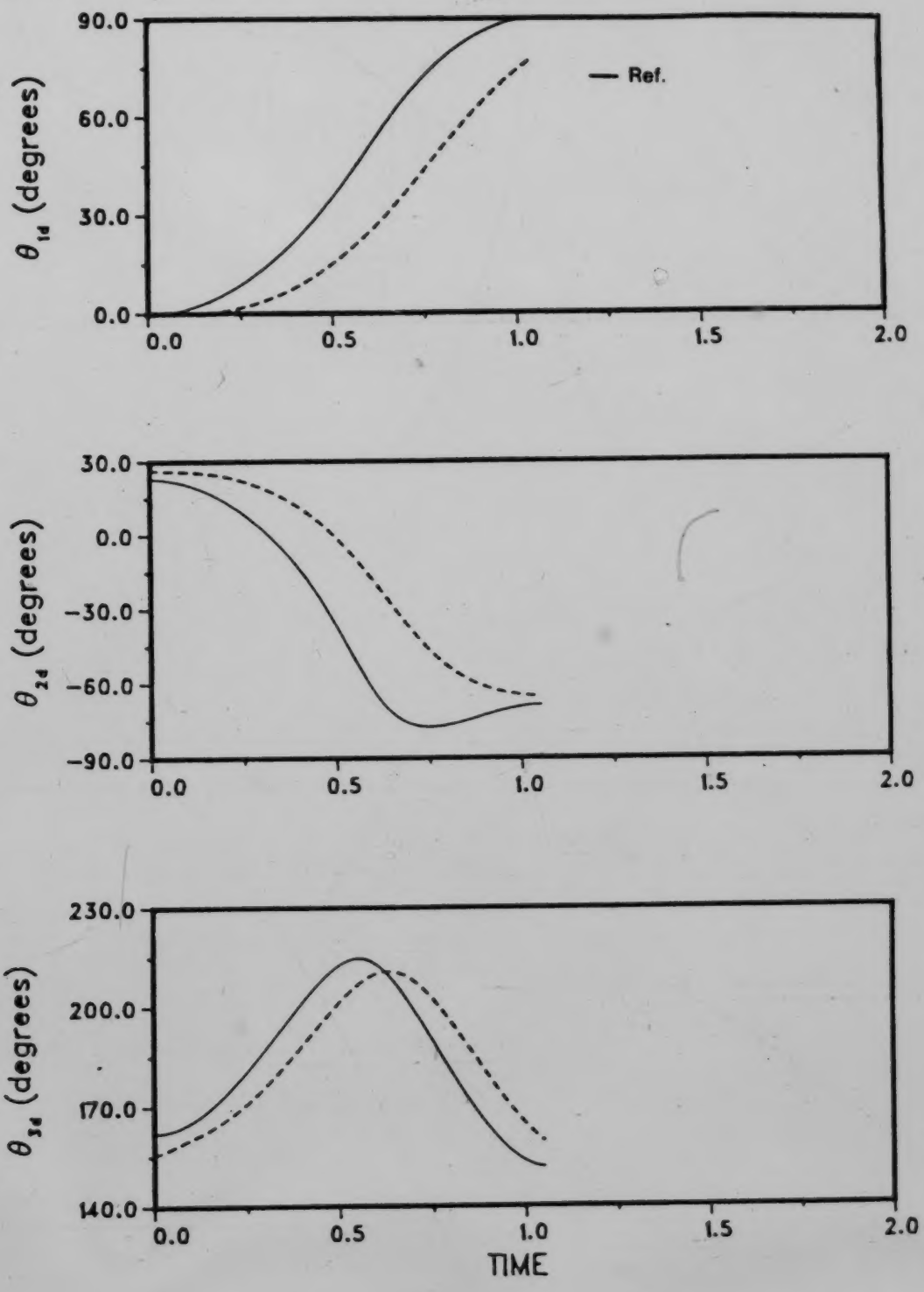


Figure 5.4 Free Motion State Space Tracking with α=0.0

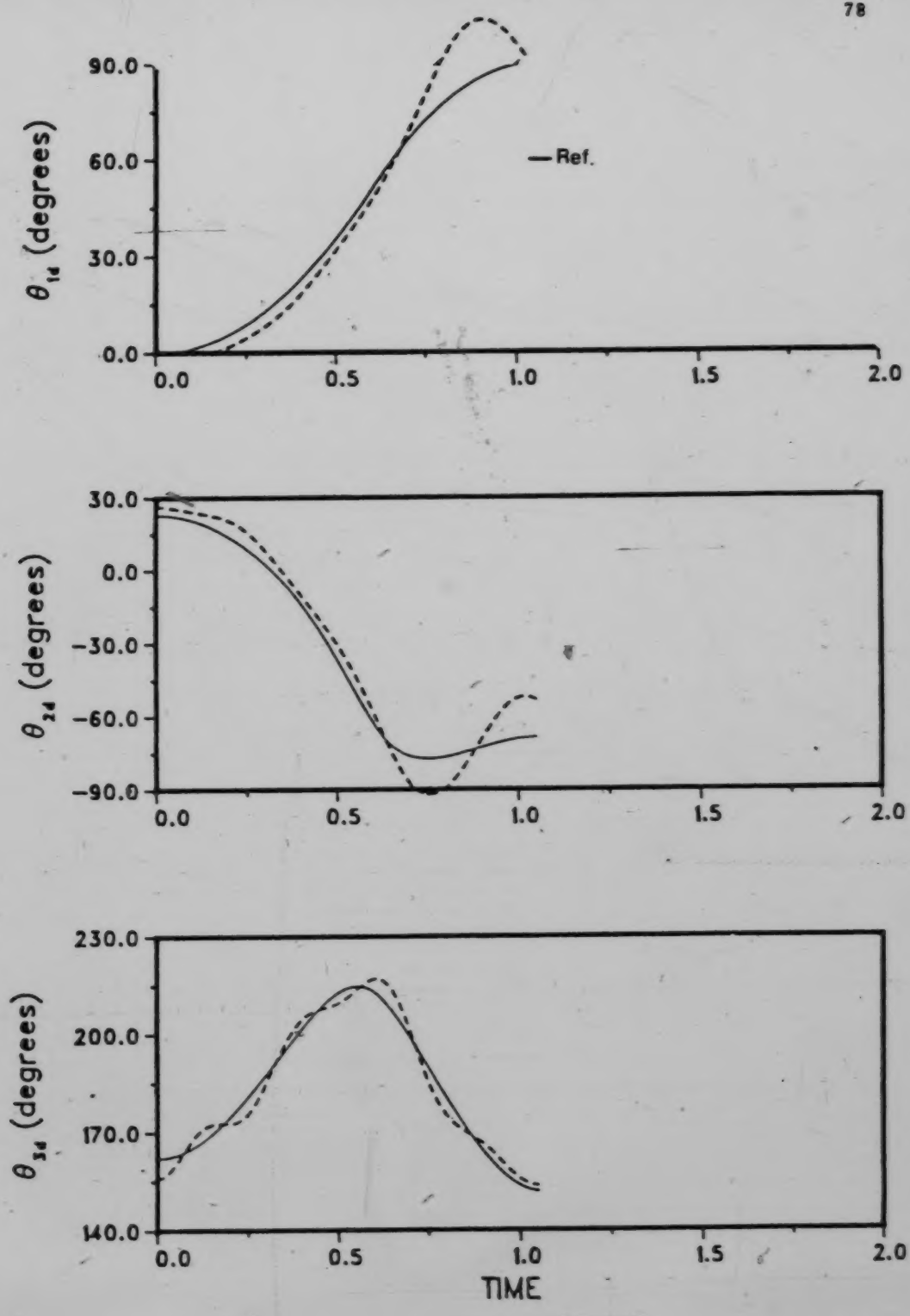


Figure 5.5 Free Motion State Space Tracking with α=1.0

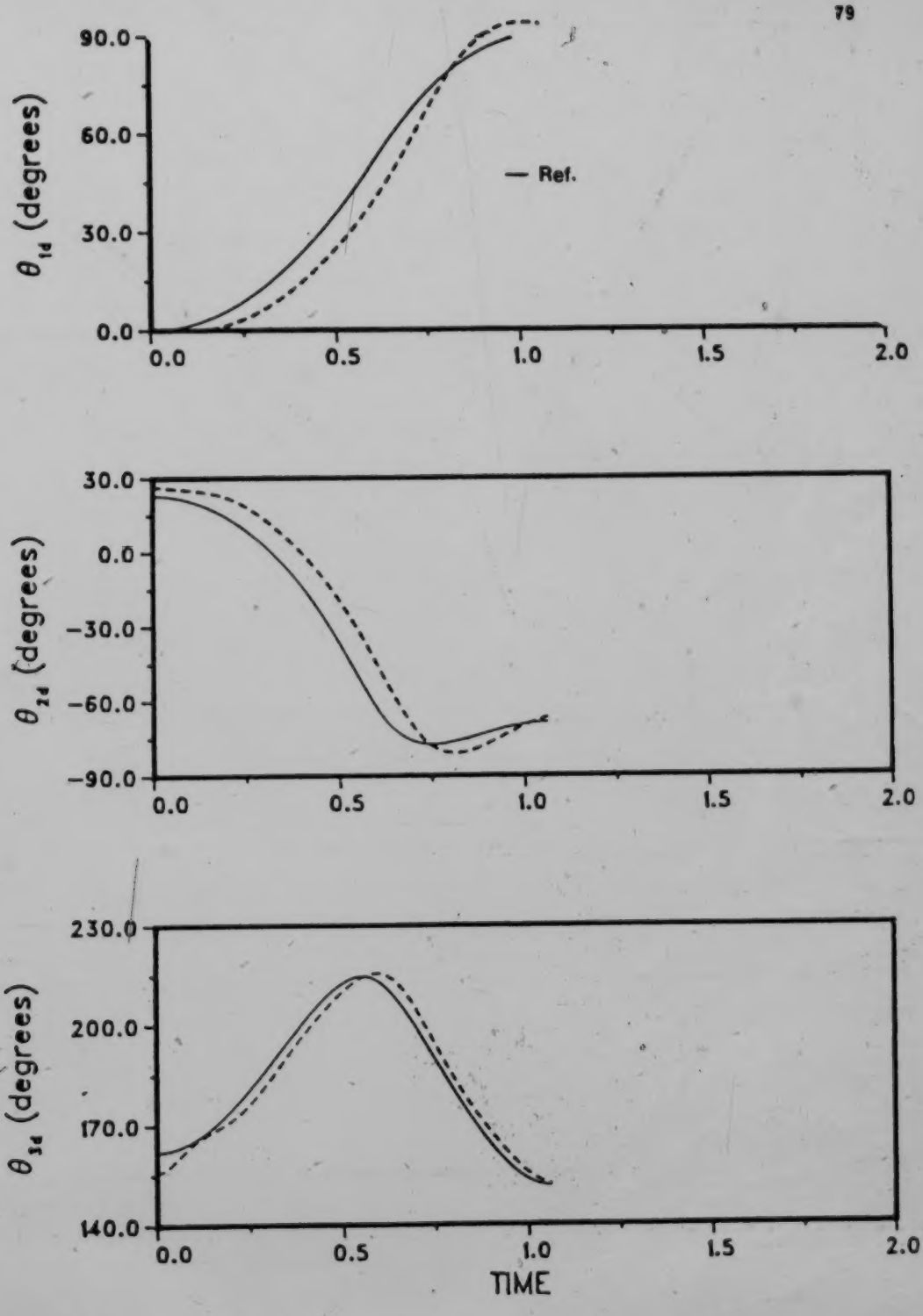


Figure 5.6 Free Motion State Space Tracking with α=0.8

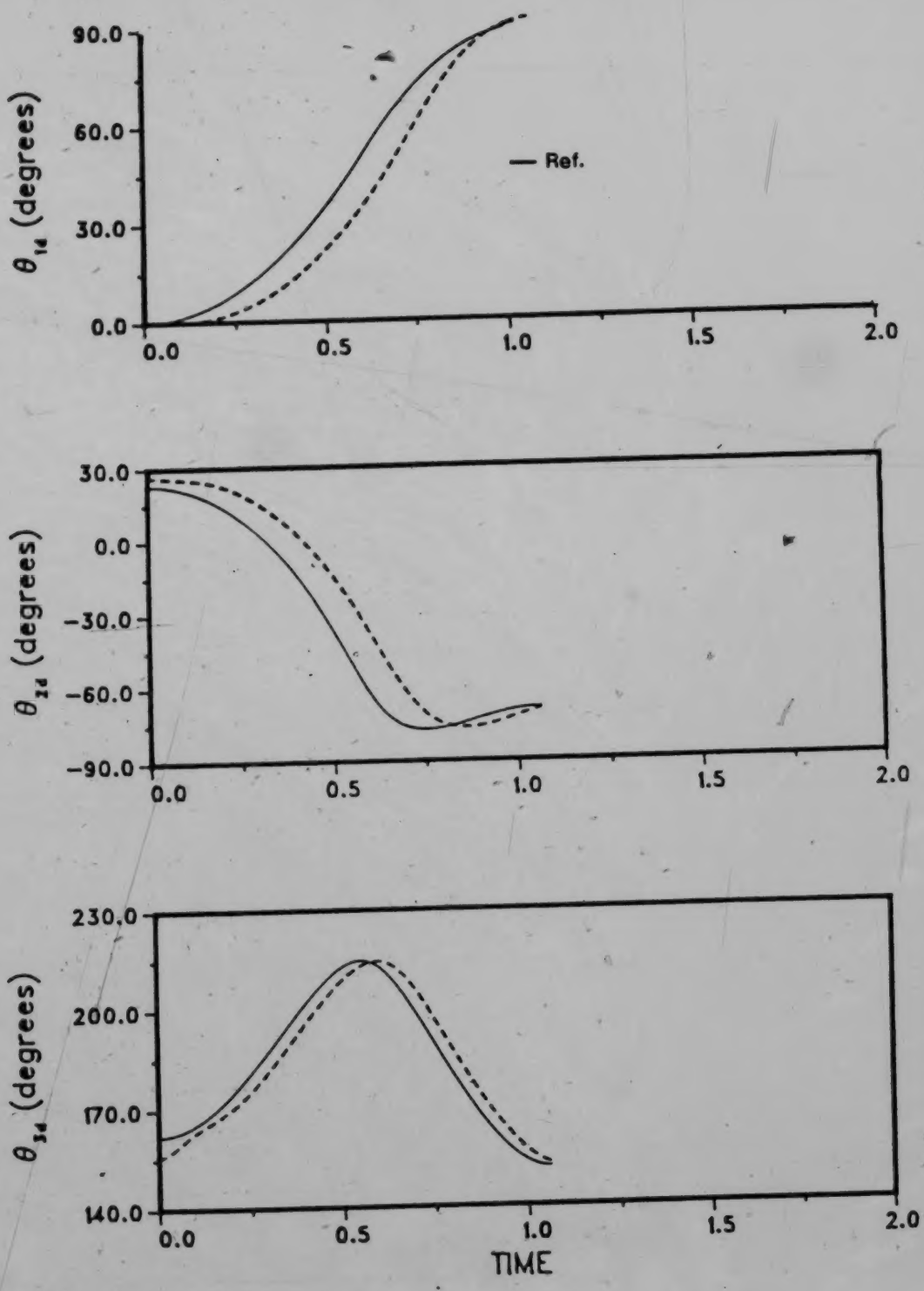


Figure 5.7 Free Motion State Space Tracking with α=0.6

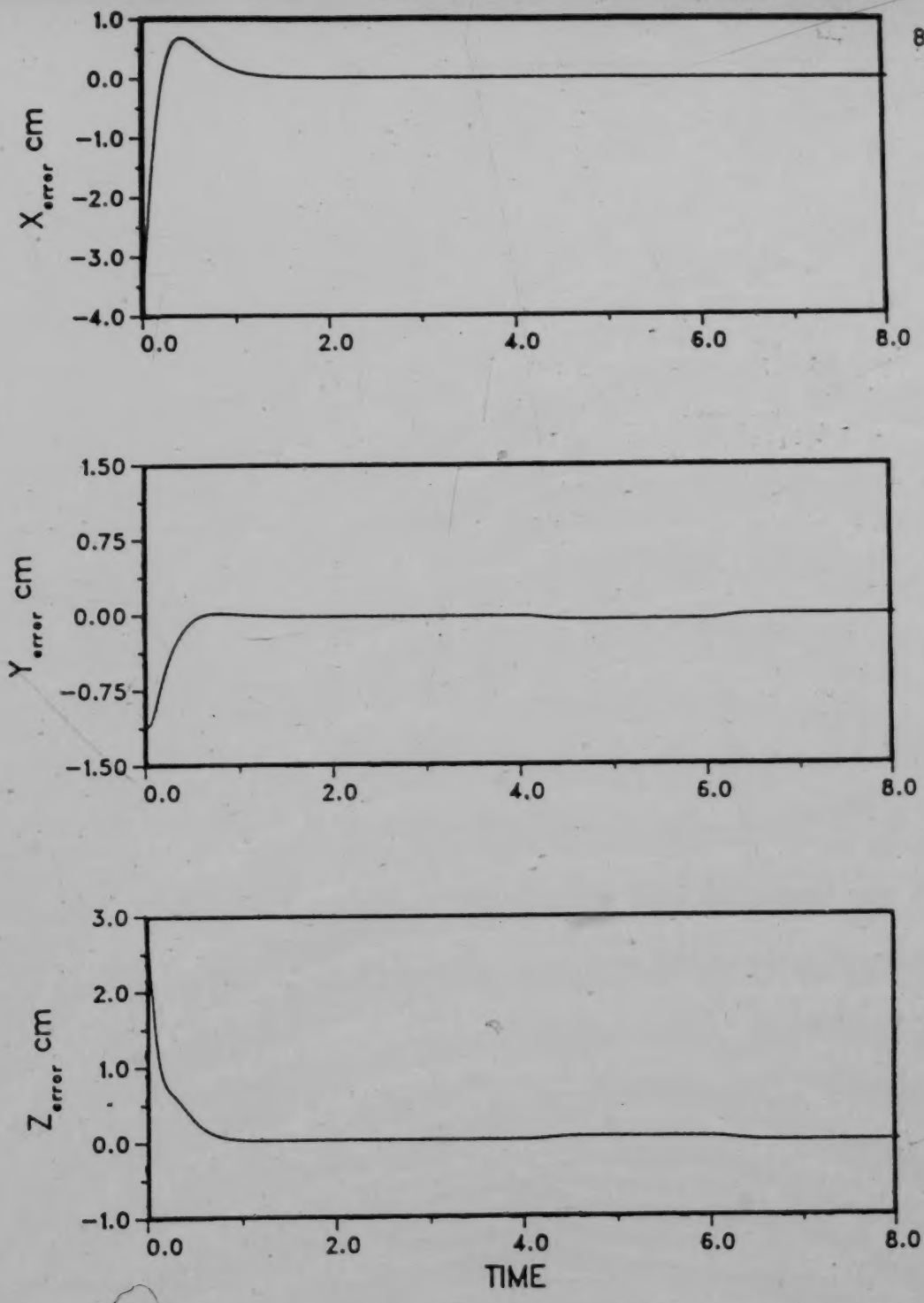


Figure 5.8 Linearized Statespace Controller Tracking Errors for two poles located with ρ = 0.25 and the third pole located at twice this value

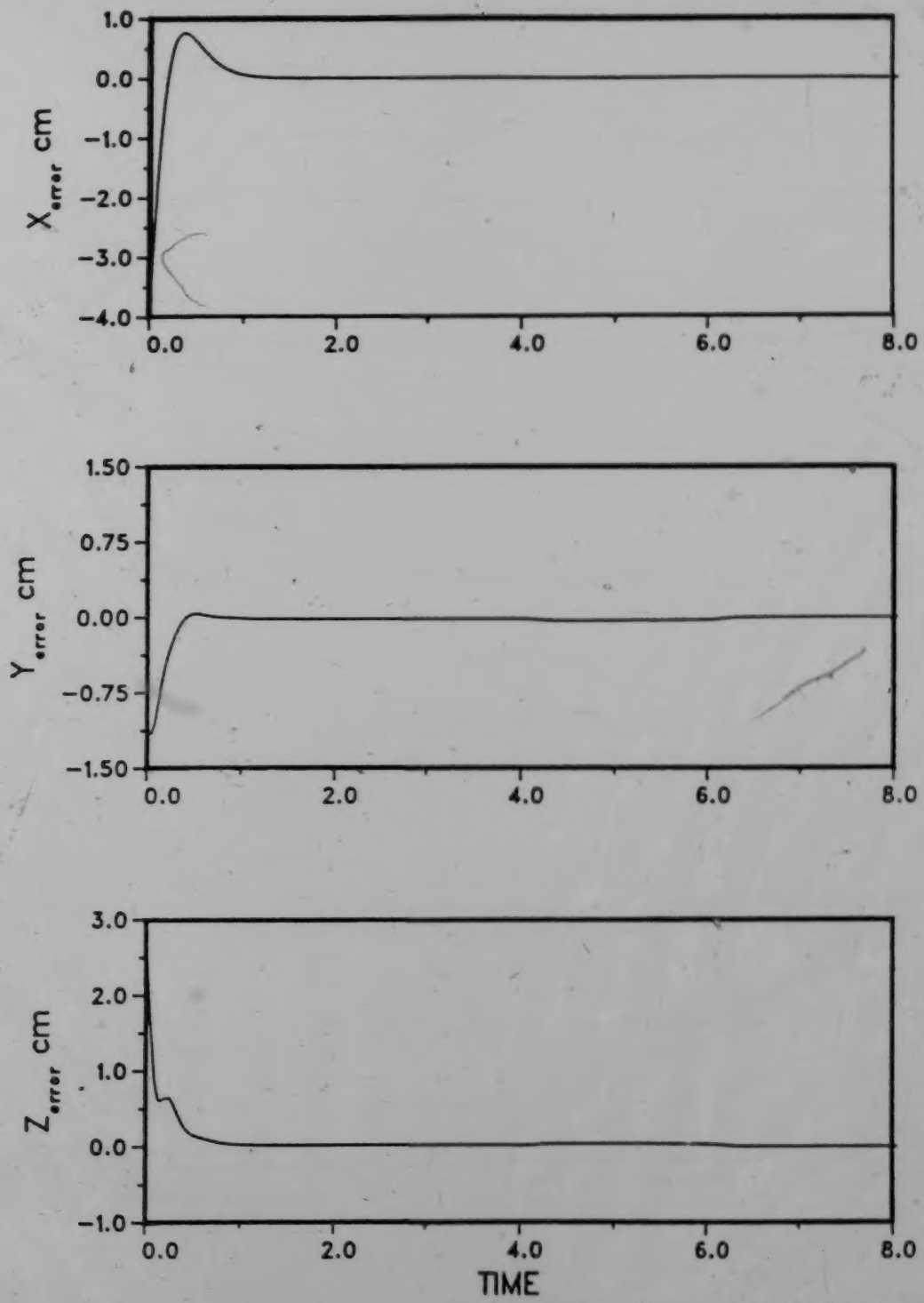


Figure 5.9 Linearized Statespace Controller Tracking Errors for two poles located with ρ = 0.20 and the third pole located at three times this value

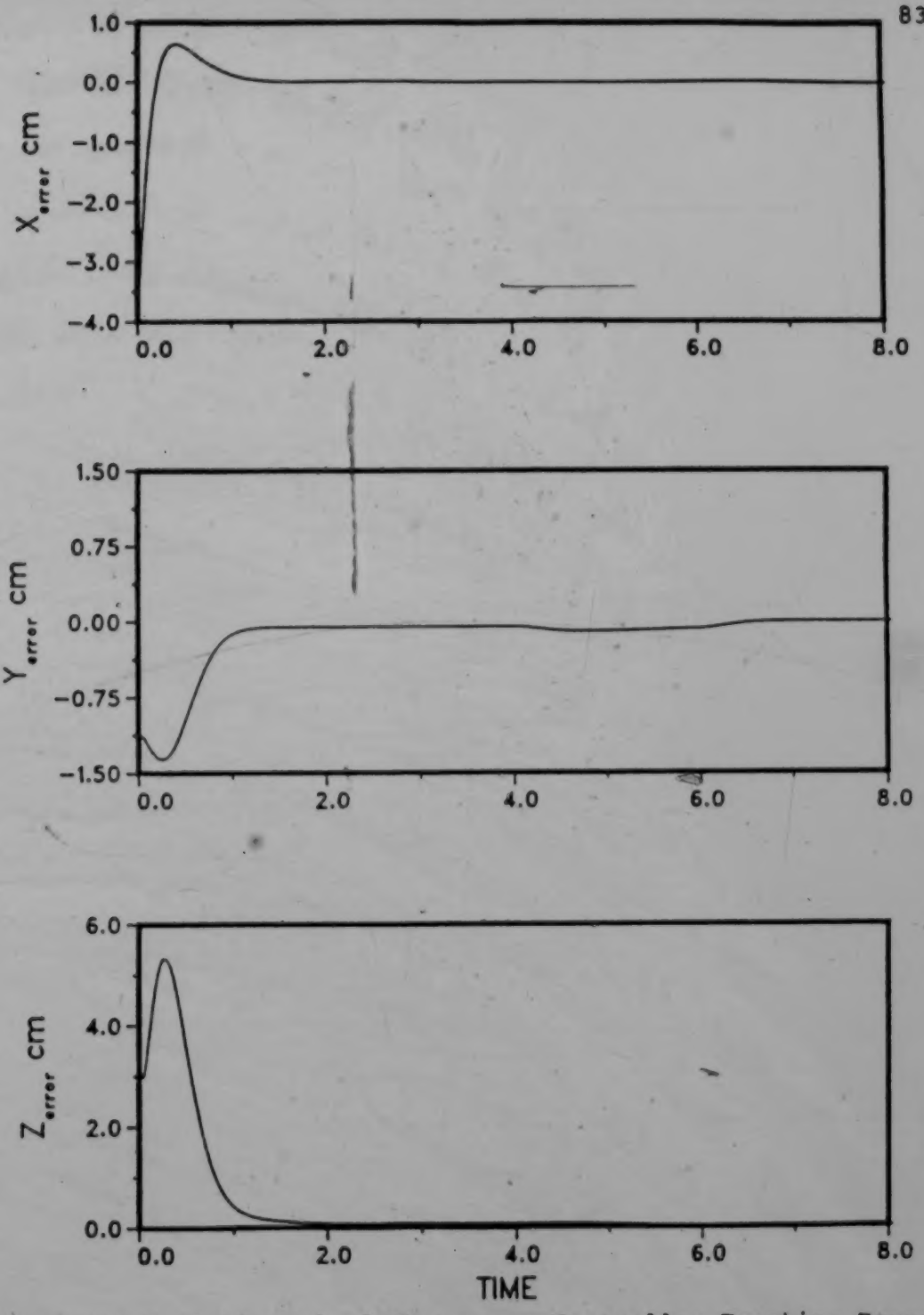


Figure 5.10 Linearized Statespace Controller Tracking Errors for two poles located at -5.0 and the third pole located at four times this value

6. Conclusions

6.1 Review of Results

Choice of a control strategy suitable for governing robotic arm motion tends to be highly task dependent. For the application considered in this thesis, free, guarded and compliant motions were all required. For all three phases, a trajectory planner was used to provide joint reference values used as inputs to the individual joint servos.

To date, the control strategy most often implemented in a robotic arm motion environment has been classical fixed gain control. Such strategies are easily understood and simple to implement. More importantly however, is that classical control is very robust and stability may be ascertained. Robustness is an important consideration when designing robotic controllers since the system parameters vary significantly with arm configuration, speed and load. Stability is vital if the arm is to operate in a safe manner.

In this thesis, a multi-phase classical PID controller was proposed and a design methodology presented. In the first level, a PD free motion controller is proposed. The design methodology accounts for both the servo-motor dynamics and the arm dynamics. The resulting controller is suitable for all ranges of motion and work loads. In the second level, an additional PI control action is incorporated in the strategy. This results in a

"split-P-mode" controller which does not have any steady state errors and is more suitable for the guarded motion phase. For the third level, a force feedback loop is cascaded with the "split-P-mode" controller so as to be able to perform compliant motions. The controllers are easily implemented in so far as the gains are all fixed (constant).

The designs represent several significant departures from designs already present in the literature. The multi-phase nature of the controller is much more suited for tasks involving three phases of motion than a single controller could be. For the final phase, the cascade system of control utilizes both force and position feedback as opposed to the hybrid controller which discards position feedback along the force controlled axes.

The state-space control designs presented, also follow this multi-phase philosophy. For the free motion phase, state-reference tracking is shown to improve the overall speed at which the motion is occuring. For the final two phases, advantage has been taken of the limited range of motion involved, and a linearized control law is developed. Utilizing the linearized system model, faster and more accurate state-space control designs have been implemented. The compliant controller also follows the cascade technique used in the classical design. As before, all the designs result in constant gain control.

In comparing the state-space controller to the conventional PID controller, two main conclusions can be

drawn. The first is that the state-space controller incorporates derivative action on the reference velocity whereas in the conventional PD controller the derivative action acted on the output alone. This places an additional zero in the closed-loop transfer function of the state-space design (compare, for example equation 5.6 with equation 4.22). The extra zero decreases the settling time and greatly increases the tracking performance. Essentially, the inclusion of θ_r allows the controller to anticipate changes in θ and provide a corresponding control action.

6.2 Future Work

Although most robot applications have relied heavily on classical control design, most of the literature has concerned itself with improved control performance based on adaptive or optimal designs. Comparisons in this area are difficult due to the lack of detailed classical designs. As well, many of the results presented in the literature are done for different robot arms operating under varying conditions and tasks. Since both optimal and adaptive control represent significant increases in implementation complexity (as well as design complexity), it is felt that a comparison of these strategies with the PID strategies presented in this work could lend significant insight into the advantages and disadvantages of each scheme.

To improve the controller presented in this work, on line identification of load parameters, which are unknown

but constant, could be carried out. Having these parameters, a functional dependency for the controller gains could be determined. With additional sensors, the transition between the free motion phase and the guarded motion phase could be carried out without a priori input from the task planner. This would require either a vision system or ultrasonic distance measurement. In general, in all of this work it was assumed that a task planner is available to perform many non-realtime functions (for example, servo references and switching between control phases was determined a priori). Future advancements in robotic technology will rely heavily on moving these tasks into real-time.

Although a multilink manipulator is highly nonlinear, fundamental linear control strategies have been shown to yield extremely accurate tracking results. Here, it is important to realize that the results presented are based on a simulated robotic model. The effects of unmodeled manipulator dynamics in robotic control has not, to date, been fully investigated.

REFERENCES

1. Brady M. N., Hollerbach J.M., Johnson T. L., Lozano-Perez T., Mason M. T., editors. Robot Motion Planning and Control (Cambridge) M.I.T. Press 1984.

2. Paul R. P., Manipulators: Mathematics, Programming and Control. (Cambridge) M.I.T. Press 1981.

3. Mujtaba, M. S., "Discussion of trajectory calculation methods", Exploratory Study of Computer Integrated Assembly Systems, Binford, T. O. et. al., Stanford University, Artificial Intelligence Laboratory, AIM 285.4, 1977.

4. Whitney, D. E., "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators", J. DSMC dec. 1972, pp 303-309

5. Lin, C. S., Chang, P. R., and Luh, J. Y. S., "Formulation and optimization of cubic polynomial joint trajectories for mechanical manipulators", Proc. IEEE conf. Decision and Control, Orlando Fl., Dec. 1982.

6. Luh, J. Y. S. and Lin, C. S., "Optimal path planning for mechanical manipulators". ASME J. DSMC Vol. 102 pp

142-151, June 1981

7. Young, K. K-D., "Control and optimization of robot trajectories", Proc. IEEE Milwaukee Symp. Auto Compensation and Control, April 1976, pp 175-178.

8. Luh J. Y. S., "Conventional controller design for industrial robots - A tutorial", IEEE Transactions on Systems Man and Cybernetics, Vol. SMC-13 pp. 298 June 1983.

9. Snyder W. E., Industrial Robots: Computer Interfacing and Control. Prentice Hall 1985.

10. Luh, J. Y. S., Walker, M. W. and Payl, R. P. C., "On line computation scheme for mechanical manipulators", J. DSMC, Vol 102 (1980) pp 69-76.

11. Walker, M. W., and Orin, D. E.. "Efficient dynamic computer simulation of robotic mechanisms", J. DSMC Vol. 102 (1982) pp 205-211.

12. Lee, C. S. G., Chung, M. J., and Lee, B. H. "Adaptive control of robot manipulators in joint and cartesian coordinates", IEEE Dec. 1983, pp 531-539.

13. Koivo, A. J. and Ten-Huei, G., "Adaptive linear

controller for robotic manipulators", IEEE Trans. on Auto. Control. Vol. AC28 Feb. 1983.

14. Chung, C. H. and Leininger, G. G., "Adaptive self-tuning control of manipulators in task and coordinate systems" IEEE 1984 pp. 546-554.

15. Snyder, W. E. and Gruver, W. A., "Microprocessor implementation of optimal control for a robot manipulator system", Proc. IEEE conf. on Decision and Control, 1979 pp 839-841.

16. Turner, T. L. and Gruver, W. A. "A viable suboptimal controller for robotic manipulators", Proc. IEEE Conference Decision and Control, Dec. 1980 pp 83-87.

17. Luo G. L., Saridis G. N., "L-Q Design of PID Controllers for Robot Arms", IEEE Journal of Robotics and Automation, Vol. RA-1, No 3, pp.152-159, Sept. 1985.

18. Applied Optimal Control, Bryson and Ho. Gimm-Blaisdell 1969.

19. Kim B. K. , Shin K. G., "Suboptimal control of industrial manipulators with a weighted minimum time fuel criterion", I.E.E.E. Tran. on Auto. Control, Vol. AC30 No. 1, 1985.

20. Luh J. Y. S., "Conventional controller design for industrial robots - A tutorial", I.E.E.E Transactions on Systems Man and Cybernetics, Vol. SMC-13 pp. 298 June 1983.

21. Raibert M. H., and Craig J. J. "Hybrid position/force control of manipulators", J. Dynamic Systems, Measurement and Control 102, pp. 126-133 June 1981.

22. Murray J. J., and Neuman C. P., "Arm: An algebraic dynamic modeling program", I.E.E.E. International Conference on Robotics, Atlanta, Georgia, pp. 103-114, 1984.

23. Neuman C. P. and Murray, J. J., "Linearization and sensitivity functions of dynamic robot models", IEEE Transactions on Systems, Man and Cybernetics, Vol 4 SMC-14, pp 805-818 December 1984.

24. Ogata, K., Modern Control Engineering, Prentice Hall, 1970.

25. Hearn, A. C., REDUCE 2 User's Manual (University of Utah) Second Edition, 1982.

26. Mitchell and Gauthier Assoc. Inc, Advanced Continuous Simulation Language (ACSL) (Concord, Mass.) 1981.

27. Luo G. L., Saridis G. N., "Robust compensation for a robotic manipulator", IEEE. Tran. on Automatic Control, Vol AC29 No. 6 1984, pp. 564-567.

28. Veitchegger, W. K., and Wu, Chi Kaur, "Robot Accuracy Analysis", Proceedings of I.E.E.E. International Conference on Systems Man and Cybernetics, Tucson Arizona, Nov. 1985, pp. 425-430.

29. Trybus, L, Rink, R. E., Gourishankar, V. G., "Design of Fast and Accurate Positional Servos for Robot Arms", Proceedings of IEEE International Conference on Systems Man and Cybernetics, Tucson Arizona, Nov. 1985, pp. 1053-1057.

30. Chae, H. A., Atkeson, C. G., Hollerbach, J. M., "Experimental Determination of the Effect of Feedforward Control on Trajectory Tracking Errors", I.E.E.E. CH2282, Feb. 1986.

31. Valavanis, K. P., Leahy, M. B., Saridis, J. N., "Real Time Evaluation of Robotic Control Methods". I.E.E.E. CH2152 July 1985.

32. Kareen, Y. Robotics for Engineers, McGraw-Hill (1985).

APPENDIX 1

Symbolic generation of the dynamic equations.

The program code used to symbolically generate the dynamic equations of motion describing the PUMA 600 is listed in this appendix. Two points should be noted about this software. The first is that in generating the system equations, there are numerous strategically placed trigonometric simplifications. In using REDUCE2, it is not possible to place these identities at the begining of the program since they will expand many of the terms. In communications with the Anthony Hearn, the primary originator of REDUCE2, this problem was pointed out. Dr. Hearn has indicated that the new version of REDUCE (REDUCE3) will still not solve this problem, although they are working on it.

The second noteworthy point is that in Neuman and Murrays original work Bejczy matrices were used for generating both the nonlinear and the linearized equations. By using Bejczy matrices, symbolic differentiation is avoided. In the REDUCE2 program, Neuman and Murray's work is faithfully followed for the non-linear equations - which is to say that Bejczy matrices are used. However, in generating the linearized equations, the REDUCE2 differentiation operator was used. Originally, ARM (Neuman and Murray's program) incorporated the Bejczy matrices so as to reduce the number of trigonometric simplifications needed. Bejczy matrices are sparse and are therefore symbolically intensive

to work with. Incorporating these matrices for the linearization procedure turned out to be beyond REDUCE2's capabilities. Neuman and Murray acknowledge that their linearization procedure takes roughly 80% of the total time used to generate both the linear and non-linear equations.

```
comment program 31.dyn;
MATRIX Q1(4,4),Q2(4,4),Q3(4,4),
       J1(4,4),J2(4,4),J3(4,4),
       A1(4,4),A2(4,4),A3(4,4),
       T01(4,4),T12(4,4),T23(4,4),
       T02(4,4),T03(4,4),T13(4,4),
       S0U11(4,4),S0U21(4,4),S0U31(4,4),S1U22(4,4),S1U32(4,4),
       S2U33(4,4),S0U22(4,4),S0U32(4,4),S0U33(4,4),S1U33(4,4),
       S0U212(4,4),S0U312(4,4),S0U313(4,4),S1U323(4,4),
       S0U323(4,4),S0U222(4,4),S0U322(4,4),S0U333(4,4),
       S1U333(4,4),
       C1(3,3),C2(3,3),C3(3,3),QD(1,3),
       GV(4,1),R1(4,1),R2(4,1),R3(4,1)$

COMMENT ORDER OUTPUT;

KORDER TH1,TH2,TH3;
ORDER  L1,L2,L3;
ORDER COS,SIN;

COMMENT DEFINE TRIG IDENTITIES;

LET -SIN(TH2)*SIN(TH3) + COS(TH2)*COS(TH3) = COS(TH2+TH3),
     COS(TH2)*SIN(TH3) + SIN(TH2)*COS(TH3) = SIN(TH2+TH3);


A1:= MAT( ( COS(TH1) ,  0        , -SIN(TH1), 0    ),
          ( SIN(TH1) ,  0        ,  COS(TH1), 0    ),
          ( 0        , -1        ,  0       , 0    ),
          ( 0        ,  0        ,  0       , 1    ))$


A2:= MAT( ( COS(TH2) , -SIN(TH2) , 0 , L2*COS(TH2) ),
          ( SIN(TH2) ,  COS(TH2) , 0 , L2*SIN(TH2) ),
          ( 0        ,  0        , 1 , D2          ),
          ( 0        ,  0        , 0 , 1           ))$


A3:= MAT( ( COS(TH3) ,  0        , SIN(TH3) , 0    ),
          ( SIN(TH3) ,  0        ,-COS(TH3) , 0    ),
          (    0     ,  1        , 0        , 0    ),
          (    0     ,  0        , 0        , 1    ))$


Q1:= MAT( ( 0 , -1 , 0 , 0 ),
```

University of Alberta

```
          ( 1 ,  0 , 0 , 0 ),
          ( 0 ,  0 , 0 , 0 ),
          ( 0 ,  0 , 0 , 0 ))$

Q2:=Q1 $
Q3:=Q2 $

J1:= MAT( ( J1XX ,  0   ,  0   ,  0  ) ,
          (  0   , J1YY ,  0   ,  0  ) ,
          (  0   ,  0   , J1ZZ , M1Z ) ,
          (  0   ,  0   , M1Z  , M1  ) )$

J2:= MAT( ( J2XX ,  0   ,  0   , M2X ) ,
          (  0   , J2YY ,  0   ,  0  ) ,
          (  0   ,  0   , J2ZZ , M2Z ) ,
          ( M2X  ,  0   , M2Z  , M2  ) )$

J3:= MAT( ( J3XX ,  0   ,  0   ,  0  ) ,
          (  0   , J3YY ,  0   ,  0  ) ,
          (  0   ,  0   , J3ZZ , M3Z ) ,
          (  0   ,  0   , M3Z  , M3  ) )$

COMMENT Define homogeneous tranformations;

T01:=A1 $
T12:=A2 $
T23:=A3 $
T02:=T01*T12;
T03:=T02*T23;
T13:=T12*T23;

COMMENT Calculation of U matices $

S0U11:=Q1*T01 $
S0U21:=Q1*T02 $
S0U31:=Q1*T03 $
S1U22:=Q2*T12 $
S1U32:=Q2*T13 $
S2U33:=Q3*T23 $
S0U22:=T01*S1U22 $
S0U32:=T01*S1U32 $
S0U33:=T02*S2U33 $
```

University of Alberta

```
S1U33:=T12*S2U33 $
S0U212:=Q1*S0U22 $
S0U312:=Q1*S0U32 $
S0U313:=Q1*S0U33 $
S1U323:=Q2*S1U33 $
S0U323:=S0U22*S2U33 $
S0U222:=T01*Q2*S1U22 $
S0U322:=T01*Q2*S1U32 $
S0U333:=T02*Q3*S2U33 $
S1U333:=T12*Q2*S2U33 $

COMMENT Calculation of D;

LET COS(TH1)**2 + SIN(TH1)**2 = 1;

D11:= TRACE( S0U11*J1*TP(S0U11) + S0U21*J2*TP(S0U21)
                                + S0U31*J3*TP(S0U31) );
D21:= TRACE( S0U21*J2*TP(S0U22) + S0U31*J3*TP(S0U32) );


LET COS(TH2)**2 + SIN(TH2)**2 = 1,

    COS(TH2)*COS(TH2+TH3) + SIN(TH2)*SIN(TH2+TH3)

                = COS(TH3),

    COS(TH2+TH3)**2 +SIN(TH2+TH3)**2 = 1;

D22:= TRACE( S1U22*J2*TP(S1U22) + S1U32*J3*TP(S1U32) );
D31:= TRACE( S0U31*J3*TP(S0U33) );
D32:= TRACE( S1U32*J3*TP(S1U33) );

LET COS(TH3)**2 + SIN(TH3)**2 = 1;

D33:= TRACE( S2U33*J3*TP(S2U33) );

COMMENT Calculation of C;

C1(1,1) := 0;
C1(1,2) := TRACE( S0U212*J2*TP(S0U21) + S0U312*J3*TP(S0U31) );
C1(1,3) := TRACE( S0U313*J3*TP(S0U31) );
C1(2,2) := TRACE( S0U222*J2*TP(S0U21) + S0U322*J3*TP(S0U31) );
C1(2,3) := TRACE( S0U323*J3*TP(S0U31) );
C1(3,3) := TRACE( S0U333*J3*TP(S0U31) );
C1(2,1) :=  C1(1,2) $
C1(3,1) :=  C1(1,3) $
C1(3,2) :=  C1(2,3) $

C2(1,1) := -C1(1,2) $
C2(1,2) := 0;
C2(1,3) := -C1(2,3) $
C2(2,2) := 0;
C2(2,3) := TRACE( S1U333*J3*TP(S1U32) );
C2(3,3) := TRACE( S1U333*J3*TP(S1U32) );
C2(2,1) := C2(1,2) $
C2(3,1) := C2(1,3) $
C2(3,2) := C2(2,3) $

C3(1,1) := -C1(1,3) $
C3(1,2) :=  C1(2,3) $
```

University of Alberta

```
C3(1,3) := 0;
C3(2,2) := -C2(2,3) $
C3(2,3) := 0;
C3(3,3) := 0;
C3(2,1) := C3(1,2) $
C3(3,1) := C3(1,3) $
C3(3,2) := C3(2,3) $

QD := MAT ( (TH1D),
            (TH2D),
            (TH3D)) $

COR1 := TP(QD)*C1*QD;
COR2 := TP(QD)*C2*QD;
COR3 := TP(QD)*C3*QD;

COMMENT Calculation of G;

R2:= MAT ( (R2X),
           ( 0 ),
           ( 0 ),
           ( 1 ) ) $

R3:= MAT ( ( 0 ),
           ( 0 ),
           (R3Z),
           ( 1 ) ) $

GV:= MAT ( ( 0 ),
           ( 0 ),
           (-G ),
           ( 0 ) ) $

G1 := 0;
G2 := -M2*TP(GV)*SOU22*R2 - M3*TP(GV)*SOU32*R3;
G3 := -M3*TP(GV)*SOU33*R3;
```

University of Alberta

```
OUT LIN;

ON FORT;

SV11 := DF(COR1,TH1D);
SV12 := DF(COR1,TH2D);
SV13 := DF(COR1,TH3D);
SV21 := DF(COR2,TH1D);
SV22 := DF(COR2,TH2D);
SV23 := DF(COR2,TH3D);
SV31 := DF(COR3,TH1D);
SV32 := DF(COR3,TH2D);
SV33 := DF(COR3,TH3D);

SC11 := DF(COR1,TH1);
SC12 := DF(COR1,TH2);
SC13 := DF(COR1,TH3);
SC21 := DF(COR2,TH1);
SC22 := DF(COR2,TH2);
SC23 := DF(COR2,TH3);
SC31 := DF(COR3,TH1);
SC32 := DF(COR3,TH2);
SC33 := DF(COR3,TH3);

SG11 := DF(G1,TH1);
SG12 := DF(G1,TH2);
SG13 := DF(G1,TH3);
SG21 := DF(G2,TH1);
SG22 := DF(G2,TH2);
SG23 := DF(G2,TH3);
SG31 := DF(G3,TH1);
SG32 := DF(G3,TH2);
SG33 := DF(G3,TH3);
CLEAR -SIN(TH2)*SIN(TH3) + COS(TH2)*COS(TH3);
CLEAR  COS(TH2)*SIN(TH3) + SIN(TH2)*COS(TH3);
CLEAR  COS(TH1)**2 + SIN(TH1)**2             ;
CLEAR  COS(TH2)**2 + SIN(TH2)**2             ;
CLEAR  COS(TH2)*COS(TH2+TH3) + SIN(TH2)*SIN(TH2+TH3);
CLEAR  COS(TH2+TH3)**2 +SIN(TH2+TH3)**2              ;
CLEAR  COS(TH3)**2 + SIN(TH3)**2                     ;

DI1 := D11*TH1DD + D21*TH2DD + D31*TH3DD ;
DI2 := D21*TH1DD + D22*TH2DD + D32*TH3DD ;
DI3 := D31*TH1DD + D32*TH2DD + D33*TH3DD ;

SD11 := DF(DI1,TH1);
SD12 := DF(DI1,TH2);
SD13 := DF(DI1,TH3);
SD21 := DF(DI2,TH1);
SD22 := DF(DI2,TH2);
SD23 := DF(DI2,TH3);
SD31 := DF(DI3,TH1);
SD32 := DF(DI3,TH2);
SD33 := DF(DI3,TH3);
```

University of Alberta

APPENDIX 2

Inverse dynamic software

Using the output of the REDUCE2 program, the inverse dynamics equations were solved for the task given in Chapter 2. In solving the inverse kinematics, both θ and $\dot{\theta}$ were explicitly generated. Both of these terms are needed in the statespace control scheme. For the inverse dynamic solution, $\ddot{\theta}$ was generated using a first backward difference in time. For a more accurate solution, a central difference approximation could be used. The inverse dynamic software is given below.

```
C PROGRAM 3L.INVDYN

      DIMENSION THV(3),THDV(3),THDDV(3),A(3,4)
      REAL J1XX,J1YY,J1ZZ,M1,M1Z,J2XX,J2YY,J2ZZ,M2Z,M2X,M2,
     +     J3XX,J3YY,J3ZZ,M3Z,M3

      REAL JAC(3,3),JACINV(3,3)

C     DEFINE FUNCTION STATEMENTS

C     CORIOLIS TORQUES

          C1(TH1,TH2,TH3,TH1D,TH2D,TH3D) =
     +         -2.*A2**2*COS(TH2)*SIN(TH2)*M3*TH2D*TH1D -
     +          2.*A2**2*COS(TH2)*SIN(TH2)*M2*TH2D*TH1D -
     +          2.*A2*COS(TH3)*M3Z*TH2D*TH1D -
     +          4.*A2*COS(TH2)*SIN(TH2)*M2X*TH2D*TH1D +
     +          2.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH3D*TH1D +
     +          4.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH2D*TH1D +
     +          A2*COS(TH2)*M3*TH2D**2*D2 +
     +          A2*COS(TH2)*M2*TH2D**2*D2 +
     +          A2*COS(TH2)*M2Z*TH2D**2 +
     +          2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH3D*TH1D +
     +          2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH2D*TH1D -
     +          2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH3D*TH1D -
     +          2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH2D*TH1D +
     +          SIN(TH2+TH3)*M3Z*TH3D**2*D2 +
     +          2.*SIN(TH2+TH3)*M3Z*TH3D*TH2D*D2 +
     +          SIN(TH2+TH3)*M3Z*TH2D**2*D2 +
     +          2.*COS(TH2)*SIN(TH2)*J2YY*TH2D*TH1D -
     +          2.*COS(TH2)*SIN(TH2)*J2XX*TH2D*TH1D +
     +          COS(TH2)*M2X*TH2D**2*D2

          C2(TH1,TH2,TH3,TH1D,TH2D,TH3D) =
     +          A2**2*COS(TH2)*SIN(TH2)*M3*TH1D**2 +
     +          A2**2*COS(TH2)*SIN(TH2)*M2*TH1D**2 +
     +          A2*COS(TH3)*M3Z*TH3D**2 +
     +          2.*A2*COS(TH3)*M3Z*TH3D*TH2D +
     +          A2*COS(TH3)*M3Z*TH1D**2 +
     +          2.*A2*COS(TH2)*SIN(TH2)*M2X*TH1D**2 -
     +          2.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D**2 -
     +          SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D**2 +
     +          SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH1D**2 -
     +          2.*SIN(TH2+TH3)*M3Z*TH3D*TH1D*D2 -
     +          COS(TH2)*SIN(TH2)*J2YY*TH1D**2 +
     +          COS(TH2)*SIN(TH2)*J2XX*TH1D**2

          C3(TH1,TH2,TH3,TH1D,TH2D,TH3D) =
     +          -A2*COS(TH3)*M3Z*TH2D**2 -
     +          A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D**2 -
     +          SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D**2 +
     +          SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH1D**2 +
     +          2.*SIN(TH2+TH3)*M3Z*TH2D*TH1D*D2

C         INERTIAL TORQUES

          D11(TH1,TH2,TH3) =
     +          A2**2*COS(TH2)**2*(M2+M3) +
     +          2.*A2*SIN(TH2+TH3)*COS(TH2)*M3Z +
```

University of Alberta

```
     +            2.*A2*COS(TH2)**2*M2X +
     +           SIN(TH2+TH3)**2*J3ZZ +
     +            COS(TH2)**2*J2XX + SIN(TH2)**2*J2YY +
     +            COS(TH2+TH3)**2*J3XX + M3*D2**2 +J3YY +
     +            M2*D2**2 + 2.*M2Z*D2 + J2ZZ +J1ZZ +J1XX

          D21(TH1,TH2,TH3) =
     +            A2*SIN(TH2)*(M3+M2)*D2 +
     +            A2*SIN(TH2)*M2Z + SIN(TH2)*M2X*D2 -
     +            COS(TH2+TH3)*M3Z*D2

          D22(TH1,TH2,TH3) =
     +            A2**2*(M3+M2) +
     +            2.*A2*SIN(TH3)*M3Z +
     +            2.*A2*M2X + J3ZZ + J3XX + J2YY + J2XX

          D31(TH1,TH2,TH3) =
     +            -COS(TH2+TH3)*M3Z*D2

          D32(TH1,TH2,TH3) =
     +            A2*SIN(TH3)*M3Z + J3ZZ +J3XX

          D33(TH1,TH2,TH3) = J3ZZ + J3XX

          D12(TH1,TH2,TH3) = D21(TH1,TH2,TH3)
          D13(TH1,TH2,TH3) = D31(TH1,TH2,TH3)
          D23(TH1,TH2,TH3) = D32(TH1,TH2,TH3)

C         GRAVATY  TORQUES

          G1(TH1,TH2,TH3) = 0.0

          G2(TH1,TH2,TH3) =
     +          -G*(A2*COS(TH2)*(M2+M3) + COS(TH2)*M2*R2X +
     +          SIN(TH2+TH3)*M3*R3Z)

          G3(TH1,TH2,TH3) =
     +            -SIN(TH2+TH3)*G*M3*R3Z

C         DATA BLOCK

C       ROBOT DATA

          D2  =  0.1495
          A2  =  0.4320
          D4  =  0.4320
          R2X = -0.216
          R3Z =  0.216

          J1XX = 0.00376
          J1YY = 0.00376
          J1ZZ = 0.01690
          M1   = 2.27
          M1Z  = 0.170

          J2XX = 0.9897
          J2YY = 0.1237
          J2ZZ = 0.1237
          M2Z  = 0.0
          M2X  = -4.153
```

University of Alberta

```
      M2    = 15.91

      J3XX = 0.0074
      J3YY = 0.0074
      J3ZZ = 0.1237
      M3Z  = 2.454
      M3   = 11.36
      MLOAD = 2.25
      M3 = M3 + MLOAD

C     TRAJECTORY DATA

      R1 = 2.246
      R2 = 1.890
      RZ = -0.45

      PHIACC =  0.6
      PHIDEC =  0.8
C     WRITE(6,600)
C600  FORMAT('Enter PHIacc PHIdec:',/,'?')
C     READ(5,*) PHIACC,PHIDEC

      PHII = -0.0785
      PHIF =  0.499

      PHIT = PHIF-PHII

      TS   = SQRT(                    PHIT/
     +            (0.5*PHIACC + 0.5*PHIACC**2/PHIDEC ) )

      TF   = PHIACC*TS/PHIDEC + TS

      PHIS = PHII + 0.5*PHIACC*TS**2

      PHIDM = PHIACC*TS

      THI = 0.0
      PI = 2.*ARSIN(1.)
      THF = PI/2

      WRITE(6,601) TS,TF
601   FORMAT('TS = ',F6.3,2X,'TF = ',F6.3)

C     GENERAL DATA
      G = 9.81
      N = 100
      DELTA = TF/(N-1)
      T = 0.0

      TH1D = 0.0
      TH2D = 0.0
      TH3D = 0.0
C
C     TEMPORAL DO LOOP
C
      DO 1000 I=1,N

C     DEFINE TRAJECTORY

      IF(T.LT.TS) PHID = PHIACC*T
```

University of Alberta

```
      IF(T.GE.TS) PHID = PHIACC*TS - PHIDEC*(T-TS)

      IF(T.LT.TS) PHI = PHII + 0.5*PHIACC*T**2
      IF(T.GE.TS) PHI = PHIS + PHIDM*(T-TS) - 0.5*PHIDEC*(T-TS)**2

      TH = THI + ( (PHII - PHI)/(PHII - PHIF) )*(THF - THI)

      THD = -( (THF - THI)/(PHII - PHIF) )*PHID

      YREF = (R1-R2*COS(PHI))*SIN(TH) + D2*COS(TH)
      XREF = (R1-R2*COS(PHI))*COS(TH) - D2*SIN(TH)
      ZREF = (RZ+R2*SIN(PHI))

      YDREF = ( R1 - R2*COS(PHI) )*COS(TH) *THD
     +          + SIN(TH)*R2*SIN(PHI)*PHID
     +          - D2*SIN(TH) *THD

      XDREF = -( R1 - R2*COS(PHI) )*SIN(TH) *THD
     +          + COS(TH)*R2*SIN(PHI) *PHID
     +          - D2*COS(TH) *THD

      ZDREF = R2*COS(PHI) *PHID

C     DEFINE THETA = f(XREF,YREF,ZREF)

      DUM11 = YREF*SQRT( XREF**2 + YREF**2 - D2**2 ) - D2*XREF
      DUM21 = XREF*SQRT( XREF**2 + YREF**2 - D2**2 ) + D2*YREF
      TH1REF = ATAN2(DUM11,DUM21)

      DUM12 = XREF**2 + YREF**2 + ZREF**2 - D4**2 -A2**2 + D2**2
      DUM22 = SQRT(4.*D4**2*A2**2 -
     + (XREF**2 + YREF**2 + ZREF**2 - D4**2 -A2**2 - D2**2)**2)
      TH3REF = ATAN2(DUM12,-DUM22)

      DUM13 = ZREF*(A2+D4*SIN(TH3REF)) +
     +       D4*COS(TH3REF)*(-SQRT(XREF**2+YREF**2-D2**2))
      DUM23 = ZREF*D4*COS(TH3REF) - (A2+D4*SIN(TH3REF))*
     +       (-SQRT(XREF**2 + YREF**2 -D2**2))
      TH2REF = -ATAN2(DUM13,DUM23)

C     WRITE(6,666) XREF,YREF,ZREF
C666  FORMAT(3F12.4)

      TH1DEG = TH1REF*180./PI
      TH2DEG = TH2REF*180./PI
      TH3DEG = TH3REF*180./PI
      IF(TH3DEG.LT.0.) TH3DEG = TH3DEG + 360.

C     DEFINE INVERSE JACOBIAN (FIRST DERIVITIVE)
      TH1 = TH1REF
      TH2 = TH2REF
      TH3 = TH3REF

      JAC(1,1) = -SIN(TH1)*(A2*COS(TH2) + D4*SIN(TH2+TH3))
     +           -D2*COS(TH1)

      JAC(1,2) =  COS(TH1)*(-A2*SIN(TH2) + D4*COS(TH2+TH3))

      JAC(1,3) = COS(TH1)*D4*COS(TH2+TH3)
```

University of Alberta

```
      JAC(2,1) = COS(TH1)*(A2*COS(TH2) + D4*SIN(TH2+TH3))
     +           - D2*SIN(TH1)

      JAC(2,2) = SIN(TH1)*(-A2*SIN(TH2) + D4*COS(TH2+TH3))

      JAC(2,3) = SIN(TH1)*D4*COS(TH2+TH3)

      JAC(3,1) = 0.0

      JAC(3,2) = -D4*SIN(TH2+TH3) - A2*COS(TH2)

      JAC(3,3) = -D4*SIN(TH2+TH3)

C     INVERT THE JACOBIAN

      THDV(1) = TH1
      THDV(2) = TH2
      THDV(3) = TH3

      DO 1 II=1,3
           DO 1 JJ=1,3
1          A(II,JJ) = JAC(II,JJ)
      A(1,4) = XDREF
      A(2,4) = YDREF
      A(3,4) = ZDREF
      CALL ELIMIN(A,THDV,3,3,4,DET,RATIO)


C     PREVIOUS VALUE
      TH1DP = TH1D
      TH2DP = TH2D
      TH3DP = TH3D

      TH1D=THDV(1)
      TH2D=THDV(2)
      TH3D=THDV(3)

      TH1DD = (TH1D - TH1DP)/DELTA
      TH2DD = (TH2D - TH2DP)/DELTA
      TH3DD = (TH3D - TH3DP)/DELTA

      WRITE(6,666) T,TH1,TH2,TH3,TH1D,TH2D,TH3D,TH1DD,TH2DD,TH3DD
666   FORMAT(4G15.5,/,15X,3G15.5,/,15X,3G15.5)

      THDDV(1) = TH1DD
      THDDV(2) = TH2DD
      THDDV(3) = TH3DD

      THDV(1) = TH1D
      THDV(2) = TH2D
      THDV(3) = TH3D

      THV(1) = TH1
      THV(2) = TH2
      THV(3) = TH3

      CALL STATEM(THV,THDV,THDDV)


C     DEFINE INVERSE DYNAMICS
```

University of Alberta

```
      W1 = D11(TH1,TH2,TH3)*TH1DD +
     +     D12(TH1,TH2,TH3)*TH2DD +
     +     D13(TH1,TH2,TH3)*TH3DD

      F1 = W1 +
     +     C1(TH1,TH2,TH3,TH1D,TH2D,TH3D) +
     +     G1(TH1,TH2,TH3)

      W2 = D21(TH1,TH2,TH3)*TH1DD +
     +     D22(TH1,TH2,TH3)*TH2DD +
     +     D23(TH1,TH2,TH3)*TH3DD

      F2 = W2 +
     +     C2(TH1,TH2,TH3,TH1D,TH2D,TH3D) +
     +     G2(TH1,TH2,TH3)

      W3 = D31(TH1,TH2,TH3)*TH1DD +
     +     D32(TH1,TH2,TH3)*TH2DD +
     +     D33(TH1,TH2,TH3)*TH3DD

      F3 = W3 +
     +     C3(TH1,TH2,TH3,TH1D,TH2D,TH3D) +
     +     G3(TH1,TH2,TH3)

C           OUTPUT BLOCK

            WRITE(1,100) T,W1,W2,W3
            WRITE(2,200) T,C1(TH1,TH2,TH3,TH1D,TH2D,TH3D),
     +                   C2(TH1,TH2,TH3,TH1D,TH2D,TH3D),
     +                   C3(TH1,TH2,TH3,TH1D,TH2D,TH3D),
            WRITE(3,300) T,G2(TH1,TH2,TH3),
     +                   G3(TH1,TH2,TH3)
            WRITE(4,400) T,F1,F2,F3

C           END TEMPORAL DO LOOP

 1000       T = T + DELTA

  100       FORMAT(4G15.5)
  200       FORMAT(4G15.5)
  300       FORMAT(3G15.5)
  400       FORMAT(4G15.5)

            STOP
            END

      SUBROUTINE ELIMIN(A,X,MEQN,NROW,NCOL,DET,RATIO)
      DIMENSION A(NROW,NCOL), X(NROW)
      NEQN=MEQN
      IF(NEQN.LE.NROW.AND.NEQN.LE.NCOL-1) GO TO 1
      WRITE(6,61)
   61 FORMAT(15X,'STOP DIMENSION ERROR IN ELIMIN')
      STOP
    1 AMEAN=0.0
      DO 2 I=1,NEQN
      DO 2 J=1,NEQN
    2 AMEAN=AMEAN+ABS(A(I,J))
      AMEAN=AMEAN/FLOAT(NEQN*NEQN)
      JMAX=NEQN+1
```

University of Alberta

```
      NEQNM1=NEQN-1
      DO 6 IEQN=1,NEQNM1
      IMIN=IEQN+1
      IMAX=IEQN
      DO 3 I=IMIN,NEQN
    3 IF(ABS(A(I,IEQN)).GT.ABS(A(IMAX,IEQN))) IMAX=I
      IF(IMAX.EQ.IEQN) GO TO 5
      DO 4 J=IEQN,JMAX
      AA=A(IEQN,J)
      A(IEQN,J)=A(IMAX,J)
    4 A(IMAX,J)=AA
    5 IF(ABS(A(IEQN,IEQN)/AMEAN).LT.1.E-60) GO TO 10
      DO 6 I=IMIN,NEQN
      FACT=A(I,IEQN)/A(IEQN,IEQN)
      DO 6 J=IMIN,JMAX
    6 A(I,J)=A(I,J)-FACT*A(IEQN,J)
      IF(ABS(A(NEQN,NEQN)/AMEAN).LT.1.E-60) GO TO 10
      X(NEQN)=A(NEQN,JMAX)/A(NEQN,NEQN)
      DO 8 L=2,NEQN
      I=NEQN+1-L
      SUM=A(I,JMAX)
      IP1=I+1
      DO 7 J=IP1,NEQN
    7 SUM=SUM-A(I,J)*X(J)
    8 X(I)=SUM/A(I,I)
      DETA=1.0
      DO 9 I=1,NEQN
    9 DETA=DETA*A(I,I)
      DET=DETA
      RATIO=DETA/AMEAN**NEQN
      RETURN
   10 DET=0.0
      WRITE(6,14) DET
   14  FORMAT(2X,'DET',' =',E20.8)
      RETURN
      END
       SUBROUTINE STATEM(TH,THD,THDD)

       DIMENSION TH(3),THD(3),THDD(3),AS1(3,3),AS2(3,3),BS1(3,3),
                 SC(3,3),SG(3,3),SV(3,3),SD(3,3),DINV(3,3),WK(3,3)

       REAL J1XX,J1YY,J1ZZ,M1,M1Z,J2XX,J2YY,J2ZZ,M2Z,M2X,M2,
      +     J3XX,J3YY,J3ZZ,M3Z,M3,L1,L2,L3


       TH1 = TH(1)
       TH2 = TH(2)
       TH3 = TH(3)

       TH1D = THD(1)
       TH2D = THD(2)
       TH3D = THD(3)

       TH1DD = THDD(1)
       TH2DD = THDD(2)
       TH3DD = THDD(3)

C      ROBOT DATA

       D2  =  0.1495
```

University of Alberta

```
      A2  =  0.4320
      D4  =  0.4320
      R2X = -0.216
      R3Z =  0.216

      J1XX = 0.00376
      J1YY = 0.00376
      J1ZZ = 0.01690
      M1   = 2.27
      M1Z  = 0.170

      J2XX = 0.9897
      J2YY = 0.1237
      J2ZZ = 0.1237
      M2Z  = 0.0
      M2X  = -4.153
      M2   = 15.91

      J3XX = 0.0074
      J3YY = 0.0074
      J3ZZ = 0.1237
      M3Z  = 2.454
      M3   = 11.36

      L2 = A2
      G = 9.81

C REDUCE2 GENERATED LINEARIZED DYNAMICS

C SV = DF(COR,THD)

        SV(1,1)=2.*(-L2**2*COS(TH2)*SIN(TH2)*M3*TH2D-L2**2*COS(TH2)
     . *SIN(TH2)*M2*TH2D-L2*COS(TH3)*M3Z*TH2D-2.*L2*COS(TH2)*SIN
     . (TH2)*M2X*TH2D+L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH3D+2.*L2*
     . COS(TH2)*COS(TH2+TH3)*M3Z*TH2D+SIN(TH2+TH3)*COS(TH2+TH3)*
     . J3ZZ*TH3D+SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH2D-SIN(TH2+TH3)*
     . COS(TH2+TH3)*J3XX*TH3D-SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH2D
     . +COS(TH2)*SIN(TH2)*J2YY*TH2D-COS(TH2)*SIN(TH2)*J2XX*TH2D)

        SV(1,2)=2.*(-L2**2*COS(TH2)*SIN(TH2)*M3*TH1D-L2**2*COS(TH2)*
     . SIN(TH2)*M2*TH1D-L2*COS(TH3)*M3Z*TH1D-2.*L2*COS(TH2)*SIN
     . (TH2)*M2X*TH1D+2.*L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D+L2*COS
     . (TH2)*M3*TH2D*D2+L2*COS(TH2)*M2*TH2D*D2+L2*COS(TH2)*
     . M2Z*TH2D+SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D-SIN(TH2+TH3)*
     . COS(TH2+TH3)*J3XX*TH1D+SIN(TH2+TH3)*M3Z*TH3D*D2+SIN(
     . TH2+TH3)*M3Z*TH2D*D2+COS(TH2)*SIN(TH2)*J2YY*TH1D-COS(TH2)*
     . SIN(TH2)*J2XX*TH1D+COS(TH2)*M2X*TH2D*D2)


        SV(1,3)=2.*(L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D+SIN(TH2+TH3)*
     . COS(TH2+TH3)*J3ZZ*TH1D-SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*
     . TH1D+SIN(TH2+TH3)*M3Z*TH3D*D2+SIN(TH2+TH3)*M3Z*TH2D*D2)


        SV(2,1)=2.*(L2**2*COS(TH2)*SIN(TH2)*M3*TH1D+L2**2*COS(TH2)*
     . SIN(TH2)*M2*TH1D+L2*COS(TH3)*M3Z*TH1D+2.*L2*COS(TH2)*SIN(
     . TH2)*M2X*TH1D-2.*L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D-SIN
     . (TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D+SIN(TH2+TH3)*COS(TH2+TH3)
     . *J3XX*TH1D-SIN(TH2+TH3)*M3Z*TH3D*D2-COS(TH2)*SIN(TH2)*J2YY*
     . TH1D+COS(TH2)*SIN(TH2)*J2XX*TH1D)
```

University of Alberta

```
      SV(2,2)=2.*L2*COS(TH3)*M3Z*TH3D


      SV(2,3)=2.*M3Z*(L2*COS(TH3)*TH3D+L2*COS(TH3)*TH2D-
     . SIN(TH2+TH3)*TH1D*D2)

      SV(3,1)=2.*(-L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D-SIN(TH2+TH3)*
     . COS(TH2+TH3)*J3ZZ*TH1D+SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*
     . TH1D+SIN(TH2+TH3)*M3Z*TH2D*D2)


      SV(3,2)=2.*M3Z*(-L2*COS(TH3)*TH2D+SIN(TH2+TH3)*TH1D*D2)


      SV(3,3)=0.


C SC(1,1) := DF(COR,TH);

      SC(1,1)=0.


      SC(1,2)=4.*L2**2*SIN(TH2)**2*M3*TH2D*TH1D+4.*L2**2*SIN(TH2)
     . **2*M2*TH2D*TH1D-2.*L2**2*M3*TH2D*TH1D-2.*L2**2*M2*TH2D*
     . TH1D-2.*L2*SIN(TH2+TH3)*COS(TH2)*M3Z*TH3D*TH1D-4.*L2*SIN(
     . TH2+TH3)*COS(TH2)*M3Z*TH2D*TH1D+8.*L2*SIN(TH2)**2*M2X*
     . TH2D*TH1D-2.*L2*SIN(TH2)*COS(TH2+TH3)*M3Z*TH3D*TH1D-4.*L2
     . *SIN(TH2)*COS(TH2+TH3)*M3Z*TH2D*TH1D-L2*SIN(TH2)*M3*
     . TH2D**2*D2-L2*SIN(TH2)*M2*TH2D**2*D2-L2*SIN(TH2)*M2Z*TH2D
     . **2-4.*L2*M2X*TH2D*TH1D-4.*SIN(TH2)**2*J2YY*TH2D*TH1D+
     . 4.*SIN(TH2)**2*J2XX*TH2D*TH1D-SIN(TH2)*M2X*TH2D**2*D2+4.*
     . COS(TH2+TH3)**2*J3ZZ*TH3D*TH1D+4.*COS(TH2+TH3)**2*J3ZZ*
     . TH2D*TH1D-4.*COS(TH2+TH3)**2*J3XX*TH3D*TH1D-4.*COS(TH2+TH3)
     . **2*J3XX*TH2D*TH1D+COS(TH2+TH3)*M3Z*TH3D**2*D2+2.*COS
     . (TH2+TH3)*M3Z*TH3D*TH2D*D2+COS(TH2+TH3)*M3Z*TH2D**2*D2-
     . 2.*J3ZZ*TH3D*TH1D-2.*J3ZZ*TH2D*TH1D+2.*J3XX*TH3D*TH1D+2.*
     . J3XX*TH2D*TH1D+2.*J2YY*TH2D*TH1D-2.*J2XX*TH2D*TH1D


      SC(1,3)=-2.*L2*SIN(TH2+TH3)*COS(TH2)*M3Z*TH3D*TH1D-4.*L2*
     . SIN(TH2+TH3)*COS(TH2)*M3Z*TH2D*TH1D+2.*L2*SIN(TH3)*M3Z*TH2D
     . *TH1D+4.*COS(TH2+TH3)**2*J3ZZ*TH3D*TH1D+4.*COS(TH2+TH3)**2*
     . J3ZZ*TH2D*TH1D-4.*COS(TH2+TH3)**2*J3XX*TH3D*TH1D-4.*
     . COS(TH2+TH3)**2*J3XX*TH2D*TH1D+COS(TH2+TH3)*M3Z*TH3D**2*D2+
     . 2.*COS(TH2+TH3)*M3Z*TH3D*TH2D*D2+COS(TH2+TH3)*M3Z*
     . TH2D**2*D2-2.*J3ZZ*TH3D*TH1D-2.*J3ZZ*TH2D*TH1D+2.*J3XX*
     . TH3D*TH1D+2.*J3XX*TH2D*TH1D

      SC(2,1)=0.

      SC(2,2)=TH1D*(-2.*L2**2*SIN(TH2)**2*M3*TH1D-2.*L2**2*SIN(TH2)
     . **2*M2*TH1D+L2**2*M3*TH1D+L2**2*M2*TH1D+2.*L2*SIN(TH2+
     . TH3)*COS(TH2)*M3Z*TH1D-4.*L2*SIN(TH2)**2*M2X*TH1D+2.*L2*
     . SIN(TH2)*COS(TH2+TH3)*M3Z*TH1D+2.*L2*M2X*TH1D+2.*SIN(TH2
     . )**2*J2YY*TH1D-2.*SIN(TH2)**2*J2XX*TH1D-2.*COS(TH2+TH3)**2*
     . J3ZZ*TH1D+2.*COS(TH2+TH3)**2*J3XX*TH1D-2.*COS(TH2+TH3
     . )*M3Z*TH3D*D2+J3ZZ*TH1D-J3XX*TH1D-J2YY*TH1D+J2XX*TH1D)

      SC(2,3)=2.*L2*SIN(TH2+TH3)*COS(TH2)*M3Z*TH1D**2-L2*SIN(TH3)*
```

University of Alberta

```
     . M3Z*TH3D**2-2.*L2*SIN(TH3)*M3Z*TH3D*TH2D-L2*SIN(TH3)*M3Z
     . *TH1D**2-2.*COS(TH2+TH3)**2*J3ZZ*TH1D**2+2.*COS(TH2+TH3)**2*
     . J3XX*TH1D**2-2.*COS(TH2+TH3)*M3Z*TH3D*TH1D*D2+J3ZZ*
     . TH1D**2-J3XX*TH1D**2


       SC(3,1)=0.

       SC(3,2)=TH1D*(L2*SIN(TH2+TH3)*COS(TH2)*M3Z*TH1D+L2*SIN(TH2)*
     . COS(TH2+TH3)*M3Z*TH1D-2.*COS(TH2+TH3)**2*J3ZZ*TH1D+2.*
     . COS(TH2+TH3)**2*J3XX*TH1D+2.*COS(TH2+TH3)*M3Z*TH2D*D2+
     . J3ZZ*TH1D-J3XX*TH1D)

       SC(3,3)=L2*SIN(TH2+TH3)*COS(TH2)*M3Z*TH1D**2+L2*SIN(TH3)*M3Z
     . *TH2D**2-2.*COS(TH2+TH3)**2*J3ZZ*TH1D**2+2.*COS(TH2+TH3)
     . **2*J3XX*TH1D**2+2.*COS(TH2+TH3)*M3Z*TH2D*TH1D*D2+J3ZZ*TH1D
     . **2-J3XX*TH1D**2

       SG(1,1)=0.

       SG(1,2)=0.

       SG(1,3)=0.

       SG(2,1)=0.

       SG(2,2)=G*(L2*SIN(TH2)*M3+L2*SIN(TH2)*M2+SIN(TH2)*M2*R2X-
     . COS(TH2+TH3)*M3*R3Z)

       SG(2,3)=-COS(TH2+TH3)*G*M3*R3Z

       SG(3,1)=0.

       SG(3,2)=-COS(TH2+TH3)*G*M3*R3Z

       SG(3,3)=-COS(TH2+TH3)*G*M3*R3Z

C DI1 := D11*TH1DD + D21*TH2DD + D31*TH3DD ;
C DI2 := D21*TH1DD + D22*TH2DD + D32*TH3DD ;
C DI3 := D31*TH1DD + D32*TH2DD + D33*TH3DD ;

C SD(1,1) := DF(DI1,TH1);

       SD(1,1)=0.

       SD(1,2)=-2.*L2**2*COS(TH2)*SIN(TH2)*M3*TH1DD-2.*L2**2*COS
     . (TH2)*SIN(TH2)*M2*TH1DD-2.*L2*SIN(TH2+TH3)*SIN(TH2)*M3Z*
     . TH1DD-4.*L2*COS(TH2)*SIN(TH2)*M2X*TH1DD+2.*L2*COS(TH2)*
     . COS(TH2+TH3)*M3Z*TH1DD+L2*COS(TH2)*M3*D2*TH2DD+L2*COS(TH2
     . )*M2*D2*TH2DD+L2*COS(TH2)*M2Z*TH2DD+2.*SIN(TH2+TH3)*COS
     . (TH2+TH3)*J3ZZ*TH1DD-2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*
     . TH1DD+SIN(TH2+TH3)*M3Z*D2*TH3DD+SIN(TH2+TH3)*M3Z*D2*TH2DD
     . +2.*COS(TH2)*SIN(TH2)*J2YY*TH1DD-2.*COS(TH2)*SIN(TH2)*
     . J2XX*TH1DD+COS(TH2)*M2X*D2*TH2DD

       SD(1,3)=2.*L2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1DD+2.*SIN(TH2+
     . TH3)*COS(TH2+TH3)*J3ZZ*TH1DD-2.*SIN(TH2+TH3)*COS(TH2+TH3)*
     . J3XX*TH1DD+SIN(TH2+TH3)*M3Z*D2*TH3DD+SIN(TH2+TH3)*M3Z*D2*
     . TH2DD
```

University of Alberta

```
      SD(2,1)=0.

      SD(2,2)=TH1DD*(L2*COS(TH2)*M3*D2+L2*COS(TH2)*M2*D2+L2*COS
     . (TH2)*M2Z+SIN(TH2+TH3)*M3Z*D2+COS(TH2)*M2X*D2)

      SD(2,3)=M3Z*(L2*SIN(TH2+TH3)*SIN(TH2)*TH3DD+2.*L2*SIN(TH2
     . +TH3)*SIN(TH2)*TH2DD+L2*COS(TH2)*COS(TH2+TH3)*TH3DD+2.*L2*
     . COS(TH2)*COS(TH2+TH3)*TH2DD+SIN(TH2+TH3)*D2*TH1DD)

      SD(3,1)=0.

      SD(3,2)=SIN(TH2+TH3)*M3Z*D2*TH1DD

      SD(3,3)=M3Z*(L2*SIN(TH2+TH3)*SIN(TH2)*TH2DD+L2*COS(TH2)*
     . COS(TH2+TH3)*TH2DD+SIN(TH2+TH3)*D2*TH1DD)

C                INERTIAL TORQUES

      D11 = A2**2*COS(TH2)**2*(M2+M3) +
     .      2.*A2*SIN(TH2+TH3)*COS(TH2)*M3Z +
     .      2.*A2*COS(TH2)**2*M2X +
     .      SIN(TH2+TH3)**2*J3ZZ +
     .      COS(TH2)**2*J2XX + SIN(TH2)**2*J2YY +
     .      COS(TH2+TH3)**2*J3XX + M3*D2**2 +J3YY +
     .      M2*D2**2 + 2.*M2Z*D2 + J2ZZ +J1ZZ +J1XX+2.

      D21 = A2*SIN(TH2)*(M3+M2)*D2 +
     .      A2*SIN(TH2)*M2Z + SIN(TH2)*M2X*D2 -
     .      COS(TH2+TH3)*M3Z*D2

      D22 = A2**2*(M3+M2) +
     .      2.*A2*SIN(TH3)*M3Z +
     .      2.*A2*M2X + J3ZZ + J3XX + J2YY + J2XX+3.

      D31 = -COS(TH2+TH3)*M3Z*D2

      D32 = A2*SIN(TH3)*M3Z + J3ZZ +J3XX

      D33 = J3ZZ + J3XX + 1.
      D12 = D21
      D13 = D31
      D23 = D32

      DETD   =  D33*D22*D11 - D33*D21*D12 - D32*D23*D11
     .        + D32*D21*D13 + D31*D23*D12 - D31*D22*D13

      DINV(1,1) = (D33*D22 - D32*D23)/DETD

      DINV(1,2) = ( - D33*D12 + D32*D13)/DETD

      DINV(1,3) = (D23*D12 - D22*D13)/DETD

      DINV(2,1) = ( - D33*D21 + D31*D23)/DETD

      DINV(2,2) = (D33*D11 - D31*D13)/DETD

      DINV(2,3) = ( - D23*D11 + D21*D13)/DETD
```

University of Alberta

```
      DINV(3,1) = (D32*D21 - D31*D22)/DETD

      DINV(3,2) = ( - D32*D11 + D31*D12)/DETD

      DINV(3,3) = (D22*D11 - D21*D12)/DETD

      DO 109 I=1,3
         DO 109 J=1,3
         AS1(I,J) = 0.0
         AS2(I,J)=0.0
109      BS1(I,J)=0.0

      DO 100 I=1,3
         DO 100 J=1,3
            DO 100 K=1,3
100      AS1(I,J) = AS1(I,J) - DINV(I,K)*SV(K,J)

      DO 101 I=1,3
         DO 101 J=1,3
101      WK(I,J) = SD(I,J) + SC(I,J) + SG(I,J)

      DO 102 I=1,3
         DO 102 J=1,3
            DO 102 K=1,3
102      AS2(I,J) = AS2(I,J) - DINV(I,K)*WK(K,J)

C     OUTPUT BLOCK

      WRITE(6,600)
600   FORMAT(/,' SV(I,J) :',/)
      DO 1 I=1,3
1     WRITE(6,610) (SV(I,J), J=1,3)
610   FORMAT(3G15.5)

      WRITE(6,601)
601   FORMAT(/,' SC(I,J) :',/)
      DO 2 I=1,3
2     WRITE(6,610) (SC(I,J), J=1,3)

      WRITE(6,602)
602   FORMAT(/,' SG(I,J) :',/)
      DO 3 I=1,3
3     WRITE(6,610) (SG(I,J), J=1,3)

      WRITE(6,603)
603   FORMAT(/,' SD(I,J) :',/)
      DO 4 I=1,3
4     WRITE(6,610) (SD(I,J), J=1,3)

      WRITE(6,604)
604   FORMAT(/,' DINV(I,J) :',/)
      DO 5 I=1,3
5     WRITE(6,610) (DINV(I,J), J=1,3)

      WRITE(6,605)
605   FORMAT(/,' AS1(I,J) :',/)
      DO 6 I=1,3
6     WRITE(6,610) (AS1(I,J), J=1,3)
```

University of Alberta

```
      WRITE(6,606)
606   FORMAT(/,' AS2(I,J) :',/)
      DO 7 I=1,3
7     WRITE(6,610) (AS2(I,J), J=1,3)

      WRITE(6,607)
607   FORMAT(/,' B1(I,J) :',/)
      DO 8 I=1,3
8     WRITE(6,610) (DINV(I,J), J=1,3)

      RETURN
      END
```

University of Alberta

APPENDIX 3

To investigate the various control strategies, the complete non-linear model including control was translated using the Advanced Continuous Simulation Language (ACSL) package. For all the simulations fourth-order Runge-Kutta methods were used. Note that ACSL has built in routines for numerically evaluating the linearized state-matrices, as well as the closed-loop eigenvalues. These results were compared with the symbolically generated state-matrices, and the designed closedloop poles. Results were all within reasonable agreement.

```
   INITIAL
      ALGORITHM IALG=4
      CINTERVAL CINT=.05
      TF = 8.0
      PI = 2.*ARSIN(1.)

   '  Robot Data  '

      CONSTANT THI=0.0, G=9.81

      CONSTANT D2=0.1495, A2=0.4320, D4=0.4320,...
               R2X=-0.216, R3Z=0.216

      CONSTANT J1XX=0.00376, J1YY=0.00376, J1ZZ=0.01690,...
               M1=2.27, M1Z=0.170

      CONSTANT J2XX=0.9897, J2YY=0.1237, J2ZZ=0.1237,...
               M2Z=0.0, M2X=-4.153, M2=15.91

      CONSTANT J3XX=0.0074, J3YY=0.0074, J3ZZ=0.7067,...
               M3Z=2.454, M3=11.36

      READ(15,100) TH1IC,TH2IC,TH3IC,...
                   TH1DIC,TH2DIC,TH3DIC
      100..FORMAT(/6G15.5)

      READ(10,101) KP1,KP2,KP3
      READ(10,101) KD1,KD2,KD3
      READ(10,102) KPG1,KIG1
      READ(10,102) KPG2,KIG2
      READ(10,102) KPG3,KIG3
      101..FORMAT(/3G15.5)
      102..FORMAT(/2G15.5)

    '  Trajectory Data  '

      X0 = -0.149
      Y0 =  0.587
      Z0 =  0.457


   END $'OF INITIAL'

   DYNAMIC
      DERIVATIVE

     '  COMMENTS                                                    '
     '  D(TH)*THDD + C(TH,THD) + G(TH) + B*THD = F(t)               '
     '  let H = [ F(t) - G(TH) - C(TH,THD) - B*THD ]                '
     '  THDD = INVERSE [ D(TH) ]*[ H ]                              '

      TH1DD = DINV11*H1 + DINV12*H2 + DINV13*H3
      TH2DD = DINV21*H1 + DINV22*H2 + DINV23*H3
      TH3DD = DINV31*H1 + DINV32*H2 + DINV33*H3

      H1 =  F1 - G1 - C1 - B1*TH1D
      H2 =  F2 - G2 - C2 - B2*TH2D
```

University of Alberta

```
H3 =  F3 - G3 - C3 - B3*TH3D

'       CORIOLIS TORQUES

C1 = -2.*A2**2*COS(TH2)*SIN(TH2)*M3*TH2D*TH1D - ...
      2.*A2**2*COS(TH2)*SIN(TH2)*M2*TH2D*TH1D - ...
      2.*A2*COS(TH3)*M3Z*TH2D*TH1D -                ...
      4.*A2*COS(TH2)*SIN(TH2)*M2X*TH2D*TH1D +       ...
      2.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH3D*TH1D + ...
      4.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH2D*TH1D + ...
      A2*COS(TH2)*M3*TH2D**2*D2 + ...
      A2*COS(TH2)*M2*TH2D**2*D2 + ...
      A2*COS(TH2)*M2Z*TH2D**2 + ...
      2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH3D*TH1D + ...
      2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH2D*TH1D - ...
      2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH3D*TH1D - ...
      2.*SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH2D*TH1D + ...
      SIN(TH2+TH3)*M3Z*TH3D**2*D2 + ...
      2.*SIN(TH2+TH3)*M3Z*TH3D*TH2D*D2 + ...
      SIN(TH2+TH3)*M3Z*TH2D**2*D2 + ...
      2.*COS(TH2)*SIN(TH2)*J2YY*TH2D*TH1D - ...
      2.*COS(TH2)*SIN(TH2)*J2XX*TH2D*TH1D + ...
      COS(TH2)*M2X*TH2D**2*D2

 C2 = A2**2*COS(TH2)*SIN(TH2)*M3*TH1D**2 + ...
      A2**2*COS(TH2)*SIN(TH2)*M2*TH1D**2 + ...
      A2*COS(TH3)*M3Z*TH3D**2 + ...
      2.*L2*COS(TH3)*M3Z*TH3D*TH2D + ...
      A2*COS(TH3)*M3Z*TH1D**2 + ...
      2.*A2*COS(TH2)*SIN(TH2)*M2X*TH1D**2 - ...
      2.*A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D**2 - ...
      SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D**2 + ...
      SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH1D**2 - ...
      2.*SIN(TH2+TH3)*M3Z*TH3D*TH1D*D2 - ...
      COS(TH2)*SIN(TH2)*J2YY*TH1D**2 + ...
      COS(TH2)*SIN(TH2)*J2XX*TH1D**2

 C3 = -A2*COS(TH3)*M3Z*TH2D**2 - ...
      A2*COS(TH2)*COS(TH2+TH3)*M3Z*TH1D**2 - ...
      SIN(TH2+TH3)*COS(TH2+TH3)*J3ZZ*TH1D**2 + ...
      SIN(TH2+TH3)*COS(TH2+TH3)*J3XX*TH1D**2 + ...
      2.*SIN(TH2+TH3)*M3Z*TH2D*TH1D*D2

'       INERTIAL TORQUES

D11 = A2**2*COS(TH2)**2*(M2+M3) + ...
      2.*A2*SIN(TH2+TH3)*COS(TH2)*M3Z + ...
      2.*A2*COS(TH2)**2*M2X +  ...
      SIN(TH2+TH3)**2*J3ZZ + ...
      COS(TH2)**2*J2XX + SIN(TH2)**2*J2YY + ...
      COS(TH2+TH3)**2*J3XX + M3*D2**2 +J3YY + ...
      M2*D2**2 + 2.*M2Z*D2 + J2ZZ +J1ZZ +J1XX+2.

D21 = A2*SIN(TH2)*(M3+M2)*D2 + ...
      A2*SIN(TH2)*M2Z + SIN(TH2)*M2X*D2 - ...
      COS(TH2+TH3)*M3Z*D2

D22 = A2**2*(M3+M2) + ...
      2.*A2*SIN(TH3)*M3Z + ...
      2.*A2*M2X + J3ZZ + J3XX + J2YY + J2XX+3.
```

University of Alberta

```
      D31 = -COS(TH2+TH3)*M3Z*D2

      D32 = A2*SIN(TH3)*M3Z + J3ZZ +J3XX

      D33 = J3ZZ + J3XX + 1.
      D12 = D21
      D13 = D31
      D23 = D32

DETD  =  D33*D22*D11 - D33*D21*D12 - D32*D23*D11...
       + D32*D21*D13 + D31*D23*D12 - D31*D22*D13

DINV11 = (D33*D22 - D32*D23)/DETD

DINV12 = ( - D33*D12 + D32*D13)/DETD

DINV13 = (D23*D12 - D22*D13)/DETD

DINV21 = ( - D33*D21 + D31*D23)/DETD

DINV22 = (D33*D11 - D31*D13)/DETD

DINV23 = ( - D23*D11 + D21*D13)/DETD

DINV31 = (D32*D21 - D31*D22)/DETD

DINV32 = ( - D32*D11 + D31*D12)/DETD

DINV33 = (D22*D11 - D21*D12)/DETD

      '       GRAVATY  TORQUES                '

      G1 = 0.0

      G2 = -G*(A2*COS(TH2)*(M2+M3) + COS(TH2)*M2*R2X + ...
           SIN(TH2+TH3)*M3*R3Z)

      G3 = -SIN(TH2+TH3)*G*M3*R3Z

      '        INTEGRATION TERMS              '

      TH1D = INTEG(TH1DD,TH1DIC)
      TH2D = INTEG(TH2DD,TH2DIC)
      TH3D = INTEG(TH3DD,TH3DIC)
      TH1  = INTEG(TH1D ,TH1IC)
      TH2  = INTEG(TH2D ,TH2IC)
      TH3  = INTEG(TH3D ,TH3IC)

      '           TRAJECTORY                   '


YREF = ...
 (Y0 + 0.02*T)*( STEP(0.0) - STEP(4.0) ) + ...
 (Y0 + 0.02*T - 0.005*(T-4.)**2)*(STEP(4.)-STEP(6.)) + ...
 (Y0+0.10)*STEP(6.)
XREF = X0
ZREF = Z0

DUM11 = YREF*SQRT( XREF**2 + YREF**2 - D2**2 ) - D2*XREF
```

University of Alberta

```
DUM21 = XREF*SQRT( XREF**2 + YREF**2 - D2**2 ) + D2*YREF
TH1REF = ATAN2(DUM11,DUM21)

DUM12 = XREF**2 + YREF**2 + ZREF**2 - D4**2 -A2**2 - D2**2
DUM22 = SQRT(4.*D4**2*A2**2 - ...
(XREF**2 + YREF**2 + ZREF**2 - D4**2 -A2**2 - D2**2)**2)
P2 = ATAN2(DUM12,-DUM22)
TH3REF = FCNSW(P2,T3REF2,T3REF1,T3REF1)
   T3REF1 = ATAN2(DUM12,-DUM22)
   T3REF2 = ATAN2(DUM12,-DUM22) + 2.*PI

DUM13 = ZREF*(A2+D4*SIN(TH3REF)) +     ...
      D4*COS(TH3REF)*(-SQRT(XREF**2+YREF**2-D2**2))
DUM23 = ZREF*D4*COS(TH3REF) - (A2+D4*SIN(TH3REF))*  ...
      (-SQRT(XREF**2 + YREF**2 -D2**2))
TH2REF = -ATAN2(DUM13,DUM23)


      '          PID  CONTROLLER                    '

E1  = TH1REF - TH1
E2  = TH2REF - TH2
E3  = TH3REF - TH3


UPI1 = KPG1*E1 + KIG1*INTEG(E1,0.0)
UPI2 = KPG2*E2 + KIG2*INTEG(E2,0.0)
UPI3 = KPG3*E3 + KIG3*INTEG(E3,0.0)

F1 = KP1*(E1 + UPI1) - KD1*TH1D
F2 = KP2*(E2 + UPI2) - KD2*TH2D
F3 = KP3*(E3 + UPI3) - KD3*TH3D


END $'OF DERIVITIVE'


   X = COS(TH1) * (A2*COS(TH2) + D4*SIN(TH2+TH3) ) ...
         - D2*SIN(TH1)

   Y = SIN(TH1) * (A2*COS(TH2) + D4*SIN(TH2+TH3) ) ...
         + D2*COS(TH1)

   Z = D4*COS(TH2+TH3) - A2*SIN(TH2)

   XERROR = XREF - X
   YERROR = YREF - Y
   ZERROR = ZREF - Z

   WRITE(13,13) T,XERROR,YERROR,ZERROR
   13..FORMAT(4G15.5)

   WRITE(12,120) T,TH1,TH2,TH3,TH1D,TH2D,TH3D
   120..FORMAT(7G15.5)


         TERMT(T.GT.TF)
```

University of Alberta

```
END $'OF OUTPUT BLOCK'
END  $'OF PROGRAM'
```


University of Alberta

CHRTC 352 (X1 - 1987) MEDICAL-MORAL PROBLEMS

Instructor: Paul Flaman, S.T.D., Assistant Professor
Office: 001, tel. 433-4461, St. Joseph's College,
NW corner basement
Office Hours on class days: Mon. [redacted] 2:30-4 p.m., and Tue. and Thur. 10:30 a.m. to 12 noon, or by appointment.

Course Objectives: This course explores some common medical-moral problems in light of the situation today and the Christian tradition (Scripture, Church teaching and theology). The presentation will be from a Catholic perspective, although some other points of view will be considered.

CONTENTS OF THE COURSE:

Introduction

1) Humanhood, Personhood, Human Rights, Priorities in Needs and Values.

2) Some Principles of Christian Decision in Medical Morals (Principles of Well-Formed Conscience; Free and Informed Consent; Moral Discrimination; Double Effect; Legitimate Cooperation; Professional Communication; Human Dignity in Community; the Common Good, Subsidiarity, and Functionalism; Totality and Integrity; Growth Through Suffering; Personalized Sexuality; Stewardship and Creativity).

3) Some Difficul Medical-Moral Problems:
a) Human Experimentation and Triage;
b) Sexuality and Reproduction (Contraception vs. Natural Family Planning, Sterilization, Artifical Insemination, *In Vitro* Fertilization, Treatment of Rape Victims, Abortion);
c) Reconstructing Human Beings (Organ Transplantation, Sexual Reassignment, Genetic Screening and Counseling, Genetic Reconstruction); and
d) Suffering and Death (Defining Death, Truth Telling to the Dying, Euthanasia, Letting Die in Peace).

Conclusion.

COURSE REQUIREMENTS:

A. Required Reading Materials:

1) Benedict M. Ashley, O.P., and Kevin D. O'Rourke, O.P., *Ethics of Health Care*. St. Louis: The Catholic Health Association of the United States, 1986.

2) John Gallagher, C.S.B., *Is the Human Embryo a Person?* Ottawa: Human Life Research Institute, 1985 (available from the instructor, $1.15).

3) About 35 pages of handouts (available from the instructor, $.85).

B. Mid-term examination - worth 20% of the course mark, Wed. Oct. 28, 1987 during the first 50 minutes of the class period (related to the assigned readings and class material covered before the mid-term);

C. Research essay - on a topic related to the course (worth 40% of the course mark and due Wed. Nov. 18, 1987) of 1750-2,000 words (7-8 pages